TECHSAR™

Experiments with Microcontrollers

P. Bhaskar
K. Malakondaiah

dreamtech PRESS

Distributed by:

WILEY

Experiments with Microcontrollers

Experiments with Microcontrollers

P. BHASKAR
Associate Professor
Dept. of Instrumentation Technology
Gulbarga University P.G. Centre, Raichur
Karnataka

K. MALAKONDAIAH
Professor
Dept. of Instrumentation Technology
Sri Krishnadevaraya University, Anantapur
Andhra Pradesh

©Copyright 2020 I.K. International Pvt. Ltd., New Delhi-110002.

This book may not be duplicated in any way without the express written consent of the publisher, except in the form of brief excerpts or quotations for the purposes of review. The information contained herein is for the personal use of the reader and may not be incorporated in any commercial programs, other books, databases, or any kind of software without written consent of the publisher. Making copies of this book or any portion for any purpose other than your own is a violation of copyright laws.

Limits of Liability/disclaimer of Warranty: The author and publisher have used their best efforts in preparing this book. The author make no representation or warranties with respect to the accuracy or completeness of the contents of this book, and specifically disclaim any implied warranties of merchantability or fitness of any particular purpose. There are no warranties which extend beyond the descriptions contained in this paragraph. No warranty may be created or extended by sales representatives or written sales materials. The accuracy and completeness of the information provided herein and the opinions stated herein are not guaranteed or warranted to produce any particulars results, and the advice and strategies contained herein may not be suitable for every individual. Neither Dreamtech Press nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

Trademarks: All brand names and product names used in this book are trademarks, registered trademarks, or trade names of their respective holders. Dreamtech Press is not associated with any product or vendor mentioned in this book.

ISBN: 978-93-89872-49-1

EISBN: 978-93-90078-71-4

Preface

The tiny microcontrollers are the most ubiquitous of all semiconductors. Microcontrollers contain CPU, RAM, ROM, I/O ports and clock. And also they contain A/D and D/A conversion facility, serial ports for I/O port expansion, powerful programmable timer facilities, etc. that makes them useful for many applications.

There is an explosion in the application of microcontrollers. Microcontrollers are used in video games, automobiles, toys, robotics, laptops, computer's mouse, keyboard, modem, fax card, sound card, battery charger, cellular phone, pager, watch, pocket recorder, calculator, etc. The revolution in embedded intelligence is driven by microcontrollers.

Therefore, the curriculum on microcontrollers and their applications is included in many courses (both in theory and practical) – B.E./B.Tech./M.Sc. (Electronics and Instrumentation) and a variety of interdisciplinary courses everywhere.

No doubt, there are several books available in the market on microcontrollers, but most of them are confined to the theoretical aspects, viz., architecture, instruction set and programming. Rarely, we find a book which deals with the practical aspects of microcontrollers that give experimental knowledge. This has motivated us to attempt this task with a novel approach.

One of the most widely produced and used microcontrollers in the world is the 8051 and is the most popular to study. Hence, the Intel 8051/89C51 microcontroller is chosen to design the experiments/projects. The basic knowledge of 8051 microcontroller, viz., architecture and its instruction set is required to understand the experiments of this book.

The book is organized into three parts Part I deals with the software experiments — addition, subtraction, multiplication, code conversion, etc. All the software programs are written in assembly language. Part II describes the hardware experiments. Some of the hardware experiments are designed with microcontroller kits and many are discussed with 89C51 microcontroller alone. Every experiment is presented in such way that an experimenter need not require additional information to perform the experiment. In most of the experiments both the assembly language and embedded 'C' programs are given. Part III deals with the complete projects. These projects are related to the applications of microcontroller either for measuring

and/or controlling a parameter. The salient features and descriptions of sensors/transducers, signal conditioning elements and actuators are provided wherever it is necessary.

We are sure that this book addresses the experimental needs of the students/researchers/teachers of various disciplines in science, engineering, and technology on microcontrollers at the graduate and post-graduate level.

We would like to express our sense of appreciation and thanks to Mr. L. Shrimanth Sudheer Kumar, Dr. Laxmikanth, Dr. Nagabhushan Katte and Mr. Immanuel J., the Research Scholars, Department of Instrumentation Technology, Gulbarga University Post-Graduate Centre, Raichur for their help in preparing this book. We express our sincere thanks to our colleague Prof. Parvathi C.S. for her encouragement. We are also thankful to our colleagues, research scholars and students of Department of Instrumentation, Sri Krishnadevaraya University, Anantapur who helped us to make this book a reality.

Our ideas are open for discussion. Please communicate your suggestions and comments for improvement of the quality and usefulness of the text.

P. Bhaskar
K. Malakondaiah

Contents

Preface *v*

Author *x*

PART - I
SOFTWARE EXPERIMENTS

1. Assembly language program (ALP) for addition of two 16-bit numbers 3
2. Assembly language program for subtraction of a 16-bit number from a 16-bit number 6
3. Assembly language program for multiplication of 16-bit number with an 8-bit number 9
4. Assembly language program for dividing a 16-bit number with an 8-bit number 12
5. Assembly language program for multiplication of a 16- bit number with a 16-bit number 16
6. Assembly language program for dividing a 16-bit number with a 16-bit number 20
7. Assembly language program to convert 5 digit unpacked BCD number into corresponding HEX number 24
8. Assembly language program to convert HEX number into corresponding BCD number 29
9. Assembly language program for arranging the given data in ascending order 33
10. Assembly language program for arranging the given data in descending order 36

PART -II
HARDWARE EXPERIMENTS

11. To interface a binary counter display with microcontroller 41
12. To interface a seven segment display with microcontroller 44
13. To interface a multiplexed display with microcontroller 48
14. i. To interface a stepper motor with microcontroller and 52
    ii. To rotate the stepper motor in clockwise and counter clockwise directions with full step (1.8°) and half step (0.9°) angles
15. i. To interface an 8-bit D/A converter MC1408 with microcontroller 57
    ii. To convert digital data into corresponding analog voltage and
    iii. To generate waveforms viz. square, rectangular, ramp and triangle
16. To construct an A/D converter using D/A converter by successive approximation technique 63
17. To interface an 8-bit A/D converter ADC0809 with microcontroller 67
18. To interface Liquid Crystal Display (LCD) module with microcontroller and display labels on it 71
19. i. To interface an 8-bit A/D converter ADC0809 with microcontroller and 77
    ii. To measure the analog voltage and display the same on LCD module
20. i. To interface a 12 bit A/D converter AD1674 with microcontroller 88
    ii. To measure the analog voltage and display it on LCD module
21. i. To interface a 16-bit A/D converter AD976 and 102
    ii. To measure the analog voltage and display same on LCD module
22. i. To interface a 12-bit dual slope A/D converter ICL7109 with microcontroller and 112
    ii. To measure the analog voltage and display the same on LCD module
23. i. To interface 4½ digit A/D converter ICL7135 with microcontroller and 125
    ii. To measure analog voltage and display on LCD module
24. To measure temperature of water bath 135
25. To measure the frequency of a TTL signal 144
26. To measure pulse width (ON time) of a TTL signal 157
27. To measure phase angle between two signals 170
28. To interface serial port MAX232 with microcontroller 179

PART -III
PROJECTS

1. Microcontroller based pH measurement system 189
2. Microcontroller based conductivity measurement system 198
3. Microcontroller based liquid level measurement system 209
4. Microcontroller board for measurement and control applications 218
5. Microcontroller based DC motor speed control system 222
6. Microcontroller based temperature control system 246

Author

P. Bhaskar obtained his M.Sc., M.Phil., and Ph.D. degrees in Instrumentation from Sri Krishnadevaraya University, Anantapur, AP, India. He joined the Department of Instrumentation Technology, Gulbarga University Post Graduate Centre, Yerigera, Raichur as Assistant Professor in 1995. He was promoted as Associate Professor in 2006. He has more than 16 years of teaching experience in the fields of Instrumentation and Electronics. He has published more than 30 research papers in the journals of national/international repute. He has presented more than 25 papers in both national/ international conferences. One Ph.D. and three M.Phil. degrees were awarded under his guidance. His research interests include scientific/analytical instrumentation, process instrumentation, embedded systems, PC based instrumentation and fuzzy control systems.

K. Malakondaiah jointed the faculty of University Science Instrumentation Centre and the Department of Instrumentation, Sri Krishnadevaraya University, Anantapur as Assistant Professor in 1979. He obtained his Ph.D. in Physics in the year 1985. He was promoted as a Professor in 2000. He has put in more than 30 years of teaching experience in the fields of Electronics and Instrumentation. He served as Head of the Department of Instrumentation and University Science Instrumentation Centre for ten years. He also served as Coordinator, Department of Electronics. He visited Royal Nepal Academy of Science and Technology, Kathmandu, Nepal to help in the establishment of Instrumentation Centre on behalf of COSTED, Chennai. He visited Khon-Kaen University, Thailand as a resource person to the training programme in Electronics for college teachers. He visited Shanghai, China to present his research paper in an international conference. He has published more than 40 research papers in both national and international journals. He has presented more than 30 research papers in national/international conferences. He has completed 2 major research projects and one minor research project. Under his research supervision 6 Ph.Ds and 10 M.Phils were awarded in the fields of Scientific/Analytical Instrumentation. During his private visit to USA recently, he visited Harward, Yale, MIT, Pittsburgh and Stanford Universities. He served as Director, Centre for Distance Education and made significant contribution for starting many courses under distance mode. He has been serving as Chairman, Board of Studies in Instrumentation for the last ten years.

PART I
Software Experiments

Experiment 1

AIM: Assembly language program (ALP) for addition of two 16-bit numbers.

EQUIPMENT: Microcontroller kit, power supply +5V/3A.

Working Procedure of Program: Load the data pointer (DPTR) with the address of the number n1. Read the high byte of n1 and store it in the register r2. Increment the address in DPTR to point to the low byte of n1, read it and store it in the register r3. Load DPTR with the address of the number n2 and read the high byte of n2 then store it in the register r4. Increment the address in DPTR to point to the low byte of n2 and read it. Now add the low byte of n2 (which is available in accumulator) with the low byte of n1 (which is available in r3) and store the result of low byte in the register r5. Then add high byte of n2 with high byte of n1 along with the carry generated from the previous addition. Then store middle byte of the result in r4. Clear accumulator and propagate the carry generated in the previous addition to accumulator. Load the DPTR with the address of the label 'result'. Store the result of high byte in the memory location labeled as 'result'. Store the middle byte of the result in the memory location 'result+1' and finally store the low byte of the result in the memory location 'result + 2'.

Procedure

1. Load the two 16-bit numbers (n1 & n2) in the four consecutive memory locations starting from the address 9100h to 9103h.
2. Load the program-1 into the program memory from the starting address 9000h.
3. Execute the program and verify the result in three consecutive memory locations 9104h (high byte), 9105h (middle byte), and 9106h (low byte).

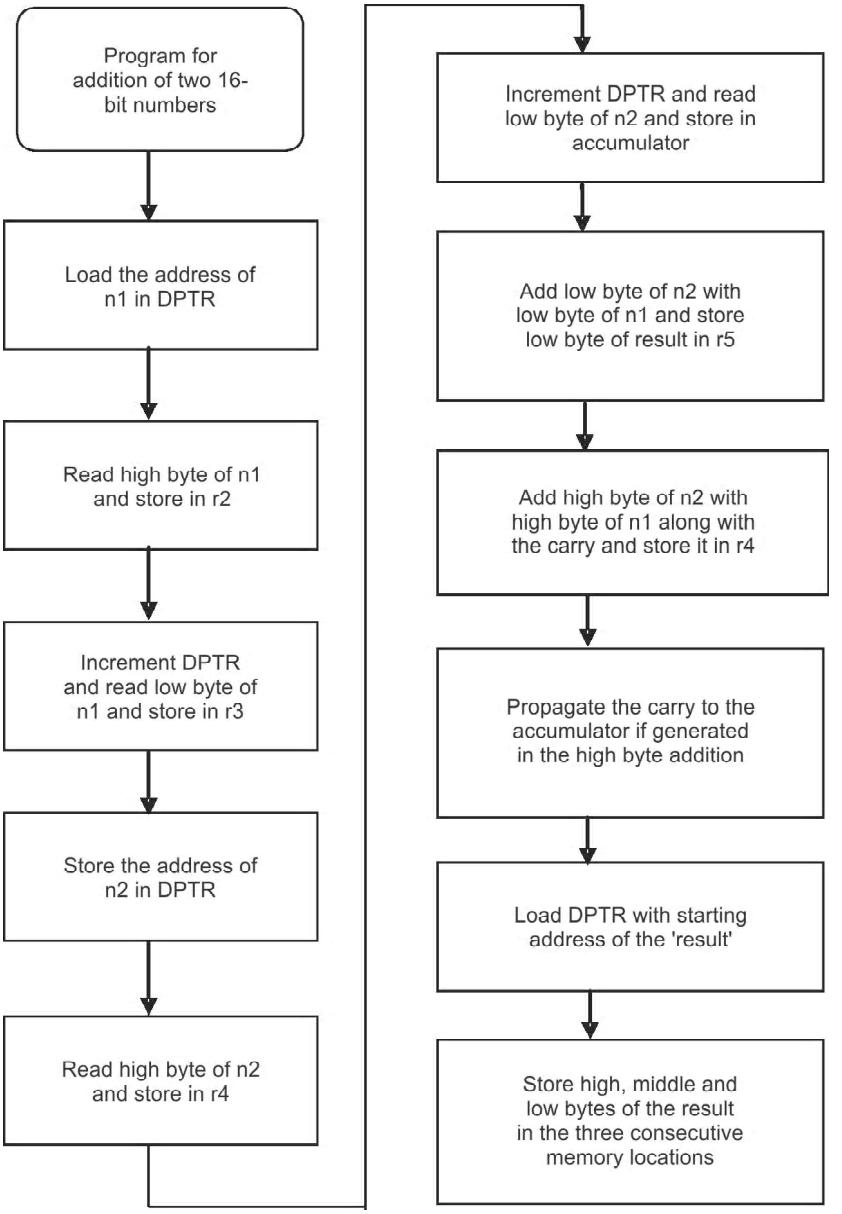

Program for addition of two 16-bit numbers
Load the address of n1 in DPTR
Read high byte of n1 and store in r2
Increment DPTR and read low byte of n1 and store in r3
Store the address of n2 in DPTR
Read high byte of n2 and store in r4
Increment DPTR and read low byte of n2 and store in accumulator
Add low byte of n2 with low byte of n1 and store low byte of result in r5
Add high byte of n2 with high byte of n1 along with the carry and store it in r4
Propagate the carry to the accumulator if generated in the high byte addition
Load DPTR with starting address of the 'result'
Store high, middle and low bytes of the result in the three consecutive memory locations

Fig. 1. Flowchart for the addition of two 16-bit numbers.

Program 1 : Program for addition of two 16-bit numbers

Address	Opcode	Label	Mnemonic		Operand	Comments
			org	9100h		
9100	30 10	n1:			db 30h,10h	;16 bit number
9102	45 60	n2:			db 45h,60h	;16 bit number
9104	00 00 00	result:			db 00h,00h,00h	;result
				;program starts here		
			org	9000h		
9000	90 91 00		mov		dptr,#n1	;load DPTR with address
9003	E0		movx		a,@dptr	;of first word n1 read high
9004	FA		mov		r2,a	;byte and store it in r2
9005	A3		inc		dptr	;increment DPTR
9006	E0		movx		a,@dptr	;read and store low byte
9007	FB		mov		r3,a	;byte in r3
9008	90 91 02		mov		dptr,#n2	;load DPTR with address
900B	E0		movx		a,@dptr	;of second word n2 read high
900C	FC		mov		r4,a	;byte and store it in r4
900D	A3		inc		dptr	;increment DPTR
900E	E0		movx		a,@dptr	;read low byte of n2
900F	2B		add		a,r3	;add low bytes n1 & n2
9010	FD		mov		r5,a	;store low byte of result in r5
9011	EC		mov		a,r4	;read high byte of n2
9012	3A		addc		a,r2	;add it with high byte of n1 ;along with carry
9013	FC		mov		r4,a	;store middle byte of result in r4
9014	E4		clr		a	;clear accumulator
9015	34 00		addc		a,#00h	;propagate carry to accumulator
9017	90 91 04		mov		dptr,#result	;load DPTR with address of result
901A	F0		movx		@dptr,a	;store high byte of result in ;memory location 'result'
901B	A3		inc		dptr	;increment DPTR
901C	EC		mov		a,r4	;read middle byte of result
901D	F0		movx		@dptr,a	;store it in the memory location ;'result+1'
901E	A3		inc		dptr	;increment DPTR
901F	ED		mov		a,r5	;read low byte of result
9020	F0		movx		@dptr,a	;store it in the memory location ;'result+2'
9021	80 FE		sjmp		$	;stay in infinite loop
			end			;end of the program

Experiment 2

AIM: Assembly language program for subtraction of a 16-bit number from a 16-bit number.

EQUIPMENT: Microcontroller kit, power supply +5V/3A.

Working Procedure of Program : Load DPTR with the address of the number n1. Read the high byte of n1 and store it in the register r2. Increment the address in DPTR to point to the low byte of n1, read it and store it in the register r3. Load DPTR with the address of the number n2 and read the high byte of n2 and store it in the register r4. Increment the address in DPTR to point to the low byte of n2 and read it. Now subtract low byte of n1 (r3) from the low byte of n2 and store the remainder in the register r5. Then subtract high byte of n1 from the high byte of n2 along with the borrow generated from the previous subtraction and store the remainder in r4. Clear accumulator and propagate the borrow generated. Load the DPTR with the address of the label 'result'. Store high byte of result in the memory location labeled as 'result' then store the middle byte of the result in the memory location 'result+1' and finally store the low byte of the result in the memory location 'result+2'. If n2 is greater than n1 result will be in the normal form else the result will be in 2's complement form.

Procedure

1. Load the two 16-bit numbers (n1 & n2) into the four consecutive memory locations starting from address 9100h to 9103h.
2. Load the program-2 into the program memory from the starting address 9000h.
3. Execute the program and verify the result in three consecutive memory locations 9104h (high byte), 9105h (middle byte), and 9106h (low byte).

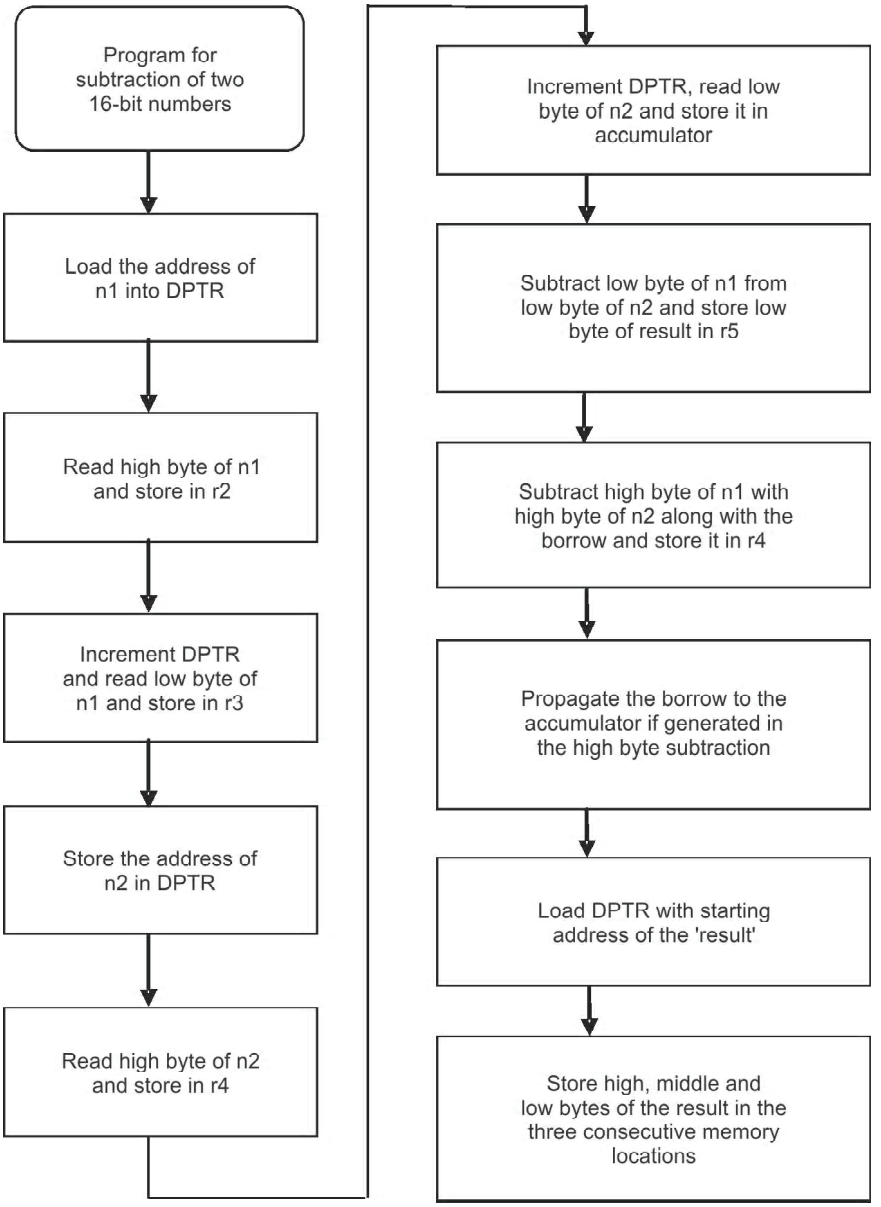

Program for subtraction of two 16-bit numbers
Load the address of n1 into DPTR
Read high byte of n1 and store in r2
Increment DPTR and read low byte of n1 and store in r3
Store the address of n2 in DPTR
Read high byte of n2 and store in r4
Increment DPTR, read low byte of n2 and store it in accumulator
Subtract low byte of n1 from low byte of n2 and store low byte of result in r5
Subtract high byte of n1 with high byte of n2 along with the borrow and store it in r4
Propagate the borrow to the accumulator if generated in the high byte subtraction
Load DPTR with starting address of the 'result'
Store high, middle and low bytes of the result in the three consecutive memory locations

Fig. 2. Flow chart for subtraction of a 16-bit number from a 16-bit number.

Program 2 : Program for subtraction of a 16-bit number from a 16-bit number

Address	Opcode	Label	Mnemonic	Operand	Comments
		org	9100h		
9100	30 10	n1:		db 30h,10h	;16-bit number
9102	45 60	n2:		db 45h,60h	;16-bit number
9104	00 00 00	result:		db 00h,00h,00h	;result
			org 9000h		
9000	90 91 00		mov	dptr,#n1	;load DPTR with address
9003	E0		movx	a,@dptr	;of n1read high
9004	FA		mov	r2,a	;byte and store it in r2
9005	A3		inc	dptr	;increment address by one
9006	E0		movx	a,@dptr	;read and store low byte
9007	FB		mov	r3,a	;in r3
9008	90 91 02		mov	dptr,#n2	;load DPTR with address of n2
900B	E0		movx	a,@dptr	;and read high byte and
900C	FC		mov	r4,a	;store it in r4
900D	A3		inc	dptr	;increment address by one
900E	E0		movx	a,@dptr	;read low byte of n2
900F	C3		clr	c	;clear carry
9010	9B		subb	a,r3	;subtract low byte of n1 from ;low byte of n2
9011	FD		mov	r5,a	;store remainder in r5
9012	EC		mov	a,r4	;read high byte of n2
9013	9A		subb	a,r2	;subtract high byte of n1 from ;high byte of n2 along with borrow
9014	FC		mov	r4,a	;store remainder in r4
9015	E4		clr	a	;clear accumulator
9016	34 00		addc	a,#00h	;propagate the borrow to acc.
9018	90 91 04		mov	dptr,#result	;load DPTR with address of ;result
901B	F0		movx	@dptr,a	;store high byte of result at the ;address 'result'
901C	A3		inc	dptr	;increment address by one
901D	EC		mov	a,r4	;read middle byte of result
901E	F0		movx	@dptr,a	;store it at the address 'result+1'
901F	A3		inc	dptr	;increment address by one
9020	ED		mov	a,r5	;read low byte of result
9021	F0		movx	@dptr,a	;store it at the address 'result+2
9022	80 FE		sjmp	$	;stay in infinite loop
			end		;end of the program

Experiment 3

AIM: Assembly language program for multiplication of a 16-bit number with an 8-bit number.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A.

Working Procedure of Program : Load DPTR with the address of multiplier n2 and store it in the register r1. Load DPTR with the address of n1 and read high byte of n1 and store it in the register r2. Then read low byte of n1 and multiply it with the multiplier (r1) and store high and low bytes of the product in r4 and r3 registers respectively. Read high byte of n1 and multiply with multiplier. Add low byte of the product with the high byte of previous product and store result in the register r4. Add the carry generated from the previous addition to the high byte of the product and store the result in the register r5. Load DPTR with the address of memory location labeled 'result' and then store high, middle and low bytes of the result into the three consecutive memory locations. The logic implemented in the program is given below.

```
      A  B  x  C
    ---------------
      B  C
+  A  C
    ---------------
   r5 r4 r3
    ---------------
```

Procedure

1. Load the multiplicand (n1) and multiplier (n2) in three consecutive memory locations starting from address 9100h to 9102h.
2. Load the program-3 into the program memory from the starting address 9000h.
3. Execute the program and verify the result in three consecutive memory locations 9103h (high byte), 9104h (middle byte) and 9105h (low byte).

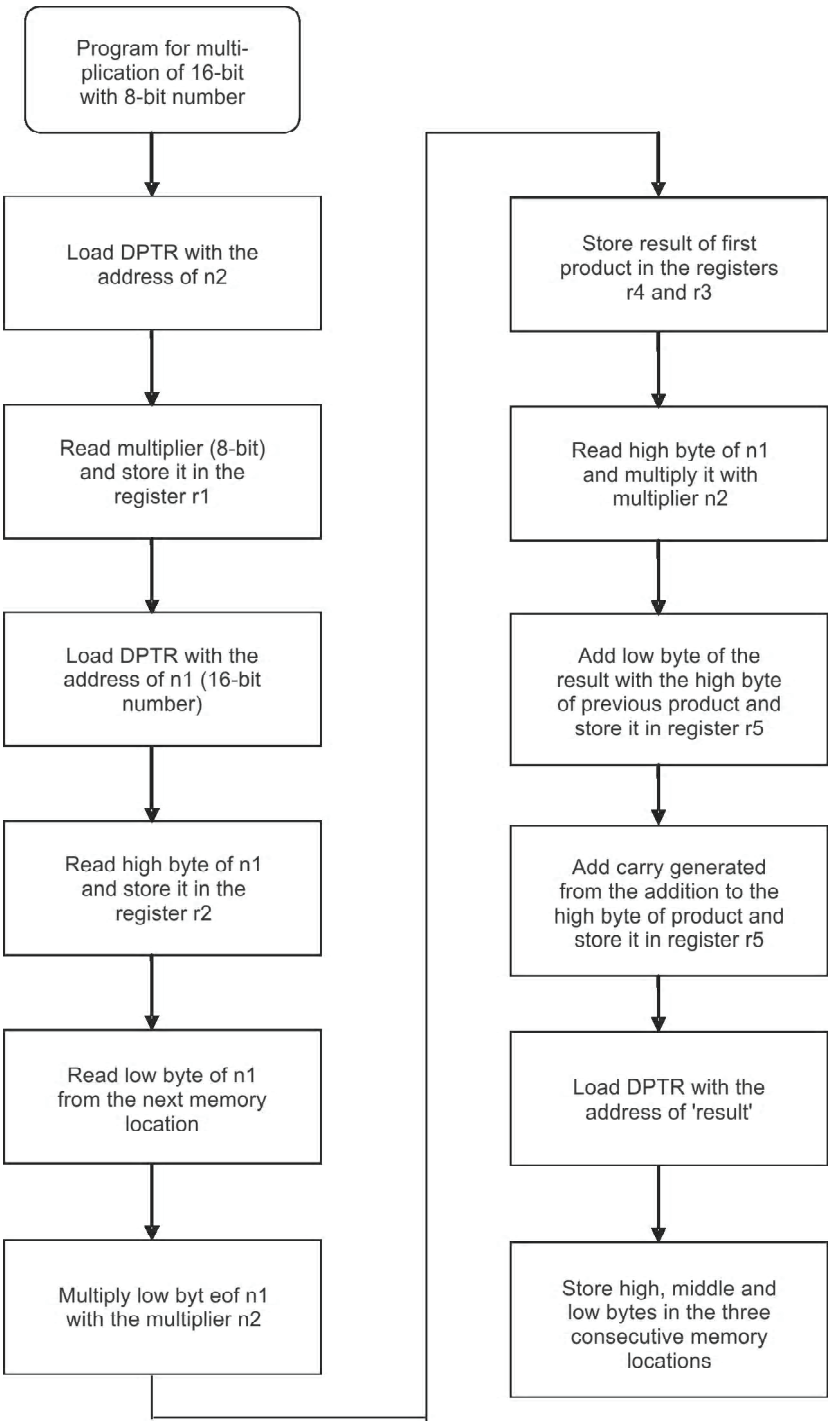

Program for multi-plication of 16-bit with 8-bit number
Load DPTR with the address of n2
Read multiplier (8-bit) and store it in the register r1
Load DPTR with the address of n1 (16-bit number)
Read high byte of n1 and store it in the register r2
Read low byte of n1 from the next memory location
Multiply low byt eof n1 with the multiplier n2
Store result of first product in the registers r4 and r3
Read high byte of n1 and multiply it with multiplier n2
Add low byte of the result with the high byte of previous product and store it in register r5
Add carry generated from the addition to the high byte of product and store it in register r5
Load DPTR with the address of 'result'
Store high, middle and low bytes in the three consecutive memory locations

Fig. 3. Flowchart for multiplication of a 16-bit number with an 8-bit number.

Program 3 : Program for multiplication of a 16-bit number with an 8-bit number

Address	Opcode	Label	Mnemonic	Operand	Comments
			org 9100h		
9100	30 10		n1:	db 30h,10h	;16-bit number
9102	80		n2:	db 80h	;8-bit number
9103	00 00 00		result:	db 00h,00h,00h	;result
			org 9000h		
9000	90 91 02		mov	dptr,#n2	;load DPTR with address
9003	E0		movx	a,@dptr	;of multiplier n2 read and
9004	F9		mov	r1,a	;store it in r1
9005	90 91 00		mov	dptr,#n1	;load DPTR with address
9008	E0		movx	a,@dptr	;of multiplicand n1 read and
9009	FA		mov	r2,a	;store high byte in r2
900A	A3		inc	dptr	;increment address pointer
900B	E0		movx	a,@dptr	;read low byte of n1
900C	89 F0		mov	b,r1	;store it in register b
900E	A4		mul	ab	;multiply low byte of n1 with n2
900F	FB		mov	r3,a	;store low byte of product in r3
9010	AC F0		mov	r4,b	;store high byte of product in r4
9012	EA		mov	a,r2	;read high byte of n1
9013	89 F0		mov	b,r1	;store n2 in register b
9015	A4		mul	ab	;multiply high byte of n1 with n2
9016	2C		add	a,r4	;add low byte of product to the high ;byte of previous product
9017	FC		mov	r4,a	;store low byte of result in r4
9018	74 00		mov	a,#00h	;load accumulator with 00
901A	35 F0		addc	a,b	;propagate carry to high byte of the ;byte of the result
901C	FD		mov	r5,a	;store high byte of result in r5
901D	90 91 03		mov	dptr,#result	;load DPTR with address of 'result'
9020	F0		movx	@dptr,a	;store high byte of the result
9021	A3		inc	dptr	;increment the address pointer
9022	EC		mov	a,r4	;read middle byte of result from r4
9023	F0		movx	@dptr,a	;store middle byte of the result
9024	A3		inc	dptr	;increment address pointer
9025	EB		mov	a,r3	;read low byte of the result from r3
9026	F0		movx	@dptr,a	;store low byte of the result
9027	80 FE		sjmp	$	;stay in infinite loop
			end		;end of the program

Experiment 4

AIM: Assembly language program for dividing a 16-bit number with an 8-bit number.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A.

Working Procedure of Program: Initialize the registers r4 and r5 with 00H for building 16-bit quotient resulting from the division. Load DPTR with the address of the divisor. Read the divisor and store it in the register r1. Load DPTR with the address of dividend, read and store it the registers r2 and r3. Subtract divisor number from the low byte of the dividend, then subtract borrow if generated from the subtraction from the high byte of the dividend. Then test whether borrow is generated from this subtraction. If borrow is not generated from the high byte subtraction, then increment the 16-bit quotient (r4:r5) by one. If borrow is generated from high byte subtraction, then the dividend is less than divisor, stop subtraction and get back the positive remainder by adding the divisor to the remainder. Store the quotient (available in r4:r5) in the locations labeled as 'result' and 'result+1'.

Procedure

1. Load the 16-bit dividend (n1) and 8-bit divider (n2) into three consecutive memory locations starting from address 9100h to 9102h.
2. Load the program-4 into the program memory from the stating address 9000h.
3. Execute the program and verify the 16-bit result (quotient) in two consecutive memory locations 9103h (high byte) and 9104h (low byte). 8-bit remainder is in r3 register.

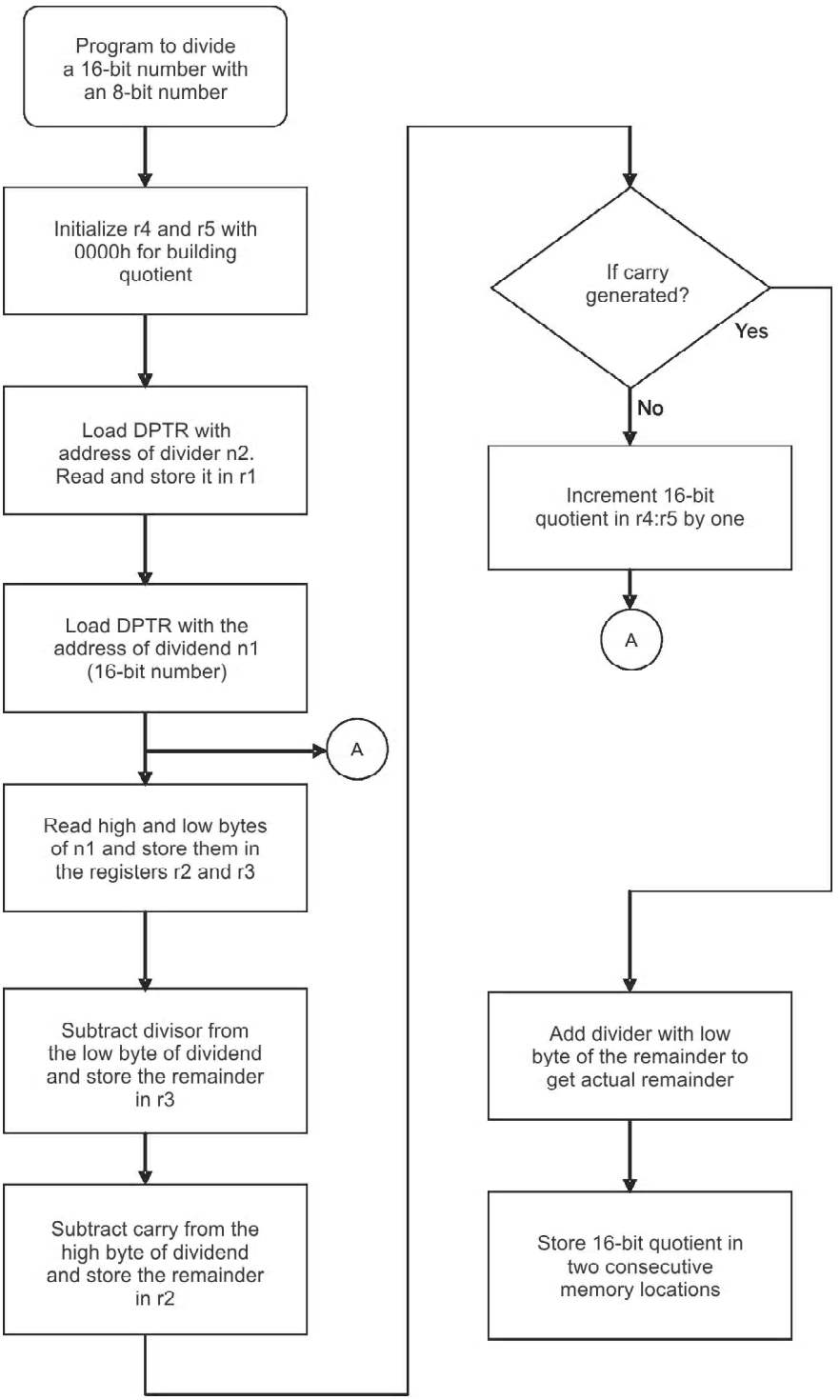

Program to divide a 16-bit number with an 8-bit number
Initialize r4 and r5 with 0000h for building quotient
Load DPTR with address of divider n2. Read and store it in r1
Load DPTR with the address of dividend n1 (16-bit number)
A
Read high and low bytes of n1 and store them in the registers r2 and r3
Subtract divisor from the low byte of dividend and store the remainder in r3
Subtract carry from the high byte of dividend and store the remainder in r2
If carry generated?
Yes
No
Increment 16-bit quotient in r4:r5 by one
A
Add divider with low byte of the remainder to get actual remainder
Store 16-bit quotient in two consecutive memory locations

Fig. 4. Flow chart to divide a 16-bit number with an 8-bit number.

Program 4 : Program for division of a 16-bit number with an 8-bit number

Address	Opcode	Label	Mnemonic		Operand	Comments
			org	9100h		
9100	30 10		n1:		db 30h,10h	;16 bit number
9102	80		n2:		db 80h	;8 bit number
9103	00 00		result:		db 00h,00h	;result
			org	9000h		
9000	7C 00		mov		r4,#00h	;clear r4 to build low byte of quotient
9002	7D 00		mov		r5,#00h	;clear r5 to build high byte of quotient
9004	90 91 02		mov		dptr,#n2	;load DPTR with address of divider n2
9007	E0		movx		a,@dptr	;read divisor
9008	F9		mov		r1,a	;store it in r1
9009	90 91 00		mov		dptr,#n1	;load DPTR with address of dividend n1
900C	E0		movx		a,@dptr	;read high byte of dividend
900D	FA		mov		r2,a	;store high byte in r2
900E	A3		inc		dptr	;increment address pointer
900F	E0		movx		a,@dptr	;read low byte of n1
9010	FB		mov		r3,a	;and store it in r3
9011	EB	x2:	mov		a,r3	;read low byte of dividend
9012	C3		clr		c	;clear carry flag
9013	99		subb		a,r1	;subtract divider from low byte of ;dividend
9014	FB		mov		r3,a	;store remainder in r3
9015	EA		mov		a,r2	;read high byte of dividend
9016	94 00		subb		a,#00h	;subtract carry from the high ;byte of dividend
9018	FA		mov		r2,a	;store remainder in r2
9019	40 0A		jc		x1	;jump if divider is greater than dividend
901B	ED		mov		a,r5	;else, increment quotient by one
901C	24 01		add		a,#01h	;increment quotient by one
901E	FD		mov		r5,a	;store it in r5
901F	EC		mov		a,r4	;read contents of r4
9020	34 00		addc		a,#00h	;propagate carry if generated ;from the from previous increment
9022	FC		mov		r4,a	;store it in r4
9023	80 EC		sjmp		x2	;repeat until dividend is greater ;than divisor
9025	EB	x1:	mov		a,r3	;if divisor is greater than dividend ;then read low byte of dividend
9026	29		add		a,r1	;add divisor to get back low ;byte of dividend
9027	FB		mov		r3,a	;store it in r3
9028	EA		mov		a,r2	;read high byte of dividend

Program 4 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
9029	34 00		addc	a,#00h	;propagate carry if generated ;to high byte ;dividend
902B	FA		mov	r2,a	;store it in r2
902C	90 91 03		mov	dptr,#result	;load DPTR with the address of result
902F	EC		mov	a,r4	;read low byte of quotient
9030	F0		movx	@dptr,a	;store it in memory
9031	A3		inc	dptr	;increment address pointer
9032	ED		mov	a,r5	;read high byte of quotient
9033	F0		movx	@dptr,a	;store it in memory
9034	80 FE		sjmp	$	;stay in infinite loop
			end		;end of the program

Experiment 5

AIM: Assembly language program for multiplication of a 16-bit number with a 16-bit number.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A.

Working Procedure of Program : Initialize the registers r7 and r6 with 0000h for building the high word of the result. Load the DPTR with the address of the number n1 (multiplicand). Read the high byte of n1 and store it in the register r0. Increment the address in DPTR to point to the low byte of n1, read and store it in the register r1. Load DPTR with the address of the number n2 (multiplier) and read the high byte of n2 and store it in the register r2. Increment the address in DPTR to point to the low byte of n2, read and store it in the register r3. Multiply low byte of n1 with low byte of n2 and store the product of high and low bytes in the registers r5 and r4 respectively. Multiply high byte of n1 with the low byte of n2, and then add low byte of product with the high byte of previous product in the previous multiplication. Store the result in the register r5 and store the high byte of the result along with the carry in the register r6. Multiply low byte of n1 with the high byte of n2, then add low byte of the product with the content of r5 and store the result in back into the register r5. Add high byte of the product to the content of r6 along with the carry. If carry generated out of this addition, then propagate it to the register r7. Multiply high byte of n1 with the high byte of n2, then add the low byte of the product to the content of r6. Then add high byte of product to r7 along with the carry and store the result back in the register r7. Store the 32 bit result in the registers r7, r6, r5 and r4 into four consecutive memory locations.

	A	B	x	C	D
			B	D	
		A	D		
		B	C		
+	A	C			
	r7	r6	r5	r4	

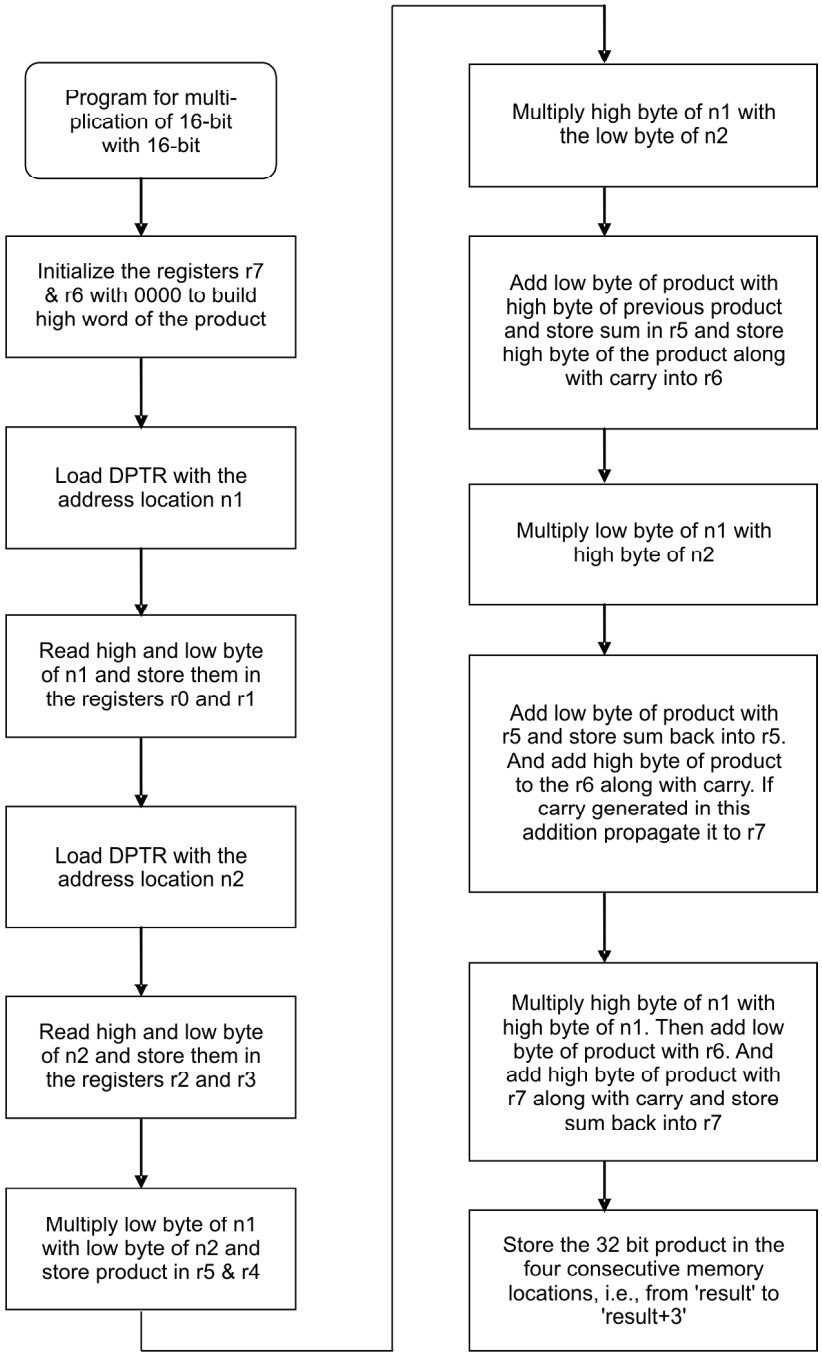

Program for multi-plication of 16-bit with 16-bit
Initialize the registers r7 & r6 with 0000 to build high word of the product
Load DPTR with the address location n1
Read high and low byte of n1 and store them in the registers r0 and r1
Load DPTR with the address location n2
Read high and low byte of n2 and store them in the registers r2 and r3
Multiply low byte of n1 with low byte of n2 and store product in r5 & r4
Multiply high byte of n1 with the low byte of n2
Add low byte of product with high byte of previous product and store sum in r5 and store high byte of the product along with carry into r6
Multiply low byte of n1 with high byte of n2
Add low byte of product with r5 and store sum back into r5. And add high byte of product to the r6 along with carry. If carry generated in this addition propagate it to r7
Multiply high byte of n1 with high byte of n1. Then add low byte of product with r6. And add high byte of product with r7 along with carry and store sum back into r7
Store the 32 bit product in the four consecutive memory locations, i.e., from 'result' to 'result+3'

Fig. 5. Flow chart for multiplication of a 16-bit number with a 16-bit number.

Procedure

1. Load the multiplicand and multiplier in four consecutive memory locations starting from address 9100h to 9103h.
2. Load the program-5 into the program memory from the starting address 9000h.
3. Execute the program and verify the result in four consecutive memory locations 9104h : 9105h (high word) and 9106h : 9107h.

Program 5 : Program for multiplication of a 16-bit number with a 16-bit number

Address	Opcode	Label	Mnemonic		Operand	Comments
9100			org	9100h		
9100	30 10		n1:		db 30h,10h	;16 bit number
9102	45 60		n2:		db 45h,60h	;16 bit number
9104	00 00		result:		db 00h,00h, 00h,00h	;result
9000			org	9000h		
9000	7F 00		mov		r7,#00h	;clear r7 and
9002	7E 00		mov		r6,#00h	;clear r6 to build high word of result
9004	90 91 00		mov		dptr,#n1	;load DPTR with address n1
9007	E0		movx		a,@dptr	;read high byte of n1
9008	F8		mov		r0,a	;store it in r0
9009	A3		inc		dptr	;increment address pointer by one
900A	E0		movx		a,@dptr	;read low byte of n1 store high
900B	F9		mov		r1,a	;store it in r1
900C	90 91 02		mov		dptr,#n2	;load DPTR with address of n2
900F	E0		movx		a,@dptr	;read high byte of n2
9010	FA		mov		r2,a	;store it in r2
9011	A3		inc		dptr	;increment address pointer by one
9012	E0		movx		a,@dptr	;read low byte of n2 and
9013	FB		mov		r3,a	;store it in r3
9014	89 F0		mov		b,r1	;store low byte of n1 in b
9016	A4		mul		ab	;multiply low byte of n1 with low ;low byte of n2
9017	FC		mov		r4,a	;store low byte of product in r4
9018	AD F0		mov		r5,b	;store high byte of product in r5
901A	E8		mov		a,r0	;store high byte of n1 in acc.
901B	8B F0		mov		b,r3	;store low byte of n2 in r3
901D	A4		mul		ab	;multiply high byte of n1 with low ;byte of n2
901E	2D		add		a,r5	;add low byte of product with high ;byte of previous product
901F	FD		mov		r5,a	;store it in r5
9020	E5 F0		mov		a,b	;store high byte of product into acc.
9022	3E		addc		a,r6	;add it to r6 along with carry
9023	FE		mov		r6,a	;store it in r6

Program 5 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
9024	50 01		jnc	m1	;don't increment r7, if carry is not ;generated in the previous addition
9026	0F		inc	r7	;else increment r7
9027	E9	m1:	mov	a,r1	;store low byte of n1 into acc.
9028	8A F0		mov	b,r2	;store high byte of n2 into b
902A	A4		mul	ab	;multiply low byte of n1 with ;high byte of n2
902B	2D		add	a,r5	;add low byte of product with r5
902C	FD		mov	r5,a	;store it in r5
902D	E5 F0		mov	a,b	;store high byte of product into acc.
902F	3E		addc	a,r6	;add it to r6 along with carry
9030	FE		mov	r6,a	;store it in r6
9031	50 01		jnc	m2	;don't increment r7, if carry is not ;in the previous addition
9033	0F		inc	r7	;else increment r7
9034	E8	m2:	mov	a,r0	;store high byte of n1 into acc.
9035	8A F0		mov	b,r2	;store high byte of n2 into b
9037	A4		mul	ab	;multiply high byte of n1 with high ;byte of n2
9038	2E		add	a,r6	;add low byte of product with r6
9039	FE		mov	r6,a	;store it in r6
903A	E5 F0		mov	a,b	;store high byte of product into acc.
903C	3F		addc	a,r7	;add it to r7 along with carry
903D	FF		mov	r7,a	;store it in r7
903E	90 91 04		mov	dptr,#result	;load DPTR with address of 'result'
9041	F0		movx	@dptr,a	;store high byte of the result
9042	A3		inc	dptr	;increment the address pointer
9043	EE		mov	a,r6	;read middle byte of result from r6
9044	F0		movx	@dptr,a	;store middle byte of the 'result+1'
9045	A3		inc	dptr	;increment address pointer
9046	ED		mov	a,r5	;read middle byte of result from r5
9047	F0		movx	@dptr,a	;store it the 'result+2'
9048	A3		inc	dptr	;increment address pointer
9049	EC		mov	a,r4	;read low byte of the result from r4
904A	F0		movx	@dptr,a	;store it the 'result+3' location
904B	80 FE		sjmp	$	;stay in infinite loop
			end		;end of the program

Experiment 6

AIM: Assembly language program for dividing a 16-bit number with a 16-bit number.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A.

Working Procedure of Program: Clear registers r4 and r5 for storing the 16-bit quotient resulting from the division. Load DPTR with the address of the divisor. Read the divisor and store it in r0 and r1 registers. Load DPTR with the address of dividend, read and store it in the registers r2 and r3. Subtract divisor from the low byte of the dividend, then subtract borrow from the high byte of the dividend. Then test whether carry flag is generated from this subtraction. If carry flag is not set from the high byte subtraction, then increment the 16 bit quotient (r4:r5) by one. If borrow is generated from high byte subtraction, then the dividend is less than divisor, stop subtraction and get back the remainder by adding divisor to the remainder. Store the quotient (available in r4:r5) in the locations labeled as 'result' and 'result+1'.

Procedure

1. Load the 16-bit dividend (n1) and 16-bit divisor (n2) into four consecutive memory locations starting from address 9100h to 9103h.
2. Load the program-6 into the program memory from the starting address 9000h.
3. Execute the program and verify the 16-bit result (quotient) in two consecutive memory locations 9104h (high byte) and 9105h (low byte). And remainder is available in r2 and r3 registers.

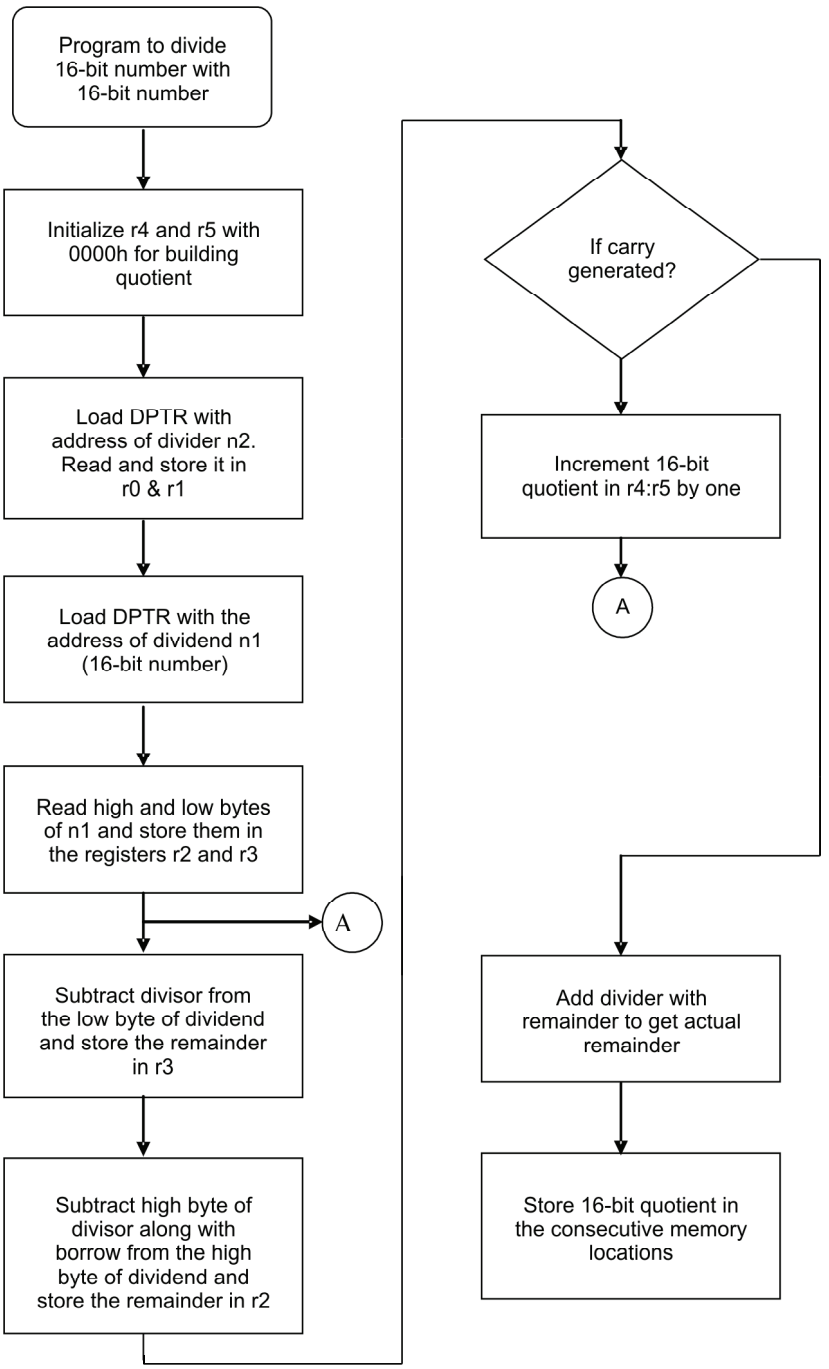

Program to divide 16-bit number with 16-bit number
Initialize r4 and r5 with 0000h for building quotient
Load DPTR with address of divider n2. Read and store it in r0 & r1
Load DPTR with the address of dividend n1 (16-bit number)
Read high and low bytes of n1 and store them in the registers r2 and r3
A
Subtract divisor from the low byte of dividend and store the remainder in r3
Subtract high byte of divisor along with borrow from the high byte of dividend and store the remainder in r2
If carry generated?
Increment 16-bit quotient in r4:r5 by one
A
Add divider with remainder to get actual remainder
Store 16-bit quotient in the consecutive memory locations

Fig. 6. Flow chart to divide a 16-bit number with a 16-bit number.

Program 6 : Program for division of a 16-bit number with a 16-bit number

Address	Opcode	Label	Mnemonic		Operand	Comments
			org	9100h		;data starts here
9100	F0 10		n1:		db 0f0h,10h	;16-bit dividend
9102	20 30		n2:		db 20h,30h	;16-bit divisor
9104	00 00		result:		db 00h,00h	;result
			org	9000h		;program starts here
9000	7C 00		mov		r4,#00h	;clear r4 to build low byte of quotient
9002	7D 00		mov		r5,#00h	;clear r5 to build high byte of quotient
9004	90 91 02		mov		dptr,#n2	;load DPTR with address
9007	E0		movx		a,@dptr	;of divider n2 read high
9008	F8		mov		r0,a	;byte store it in r0
9009	A3		inc		dptr	;increment DPTR
900A	E0		movx		a,@dptr	;read low byte of n2
900B	F9		mov		r1,a	;store it in r1
900C	90 91 00		mov		dptr,#n1	;load DPTR with address
900F	E0		movx		a,@dptr	;of dividend n1 read and
9010	FA		mov		r2,a	;store high byte in r2
9011	A3		inc		dptr	;increment DPTR
9012	E0		movx		a,@dptr	;read low byte of n1
9013	FB		mov		r3,a	;and store it in r3
9014	EB	x2:	mov		a,r3	;read low byte of dividend
9015	C3		clr		c	;clear carry flag
9016	99		subb		a,r1	;subtract divisor from low byte of ;dividend
9017	FB		mov		r3,a	;store remainder in r3
9018	EA		mov		a,r2	;read high byte of dividend
9019	98		subb		a,r0	;subtract high byte of divisor ;from the high byte of dividend ;along with borrow
901A	FA		mov		r2,a	;store remainder in r2
901B	40 0A		jc		x1	;jump to the location 'x' if ;divisor is greater than dividend
901D	ED		mov		a,r5	;else, increment low byte of ;quotient by one
901E	24 01		add		a,#01h	;increment quotient by one
9020	FD		mov		r5,a	;store it in r5
9021	EC		mov		a,r4	;read contents of r4
9022	34 00		addc		a,#00h	;propagate carry if generated ;from the from previous increment
9024	FC		mov		r4,a	;store it in r4
9025	80 ED		sjmp		x2	;repeat until dividend is less ;than divisor
9027	EB	x1:	mov		a,r3	;if divider is greater than dividend ;then read low byte of dividend
9028	29		add		a,r1	;add divisor to remainder get

Program 6 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
					;back low byte of positive dividend
9029	FB		mov	r3,a	;store it in r3
902A	EA		mov	a,r2	;read high byte of dividend
902B	38		addc	a,r0	;propagate carry if generated ;to high byte of dividend
902C	FA		mov	r2,a	;store it in r2
902D	90 91 04		mov	dptr,#result	;load DPTR with the address ;of result
9030	EC		mov	a,r4	;read low byte of quotient
9031	F0		movx	@dptr,a	;store it in memory
9032	A3		inc	dptr	;increment address pointer
9033	ED		mov	a,r5	;read high byte of quotient
9034	F0		movx	@dptr,a	;store it in memory
9035	80 FE		sjmp	$	;stay in infinite loop
			end		;end of the program

Experiment 7

AIM: Assembly language program to convert 5-digit unpacked BCD number into corresponding HEX number.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A.

Working Procedure of Program: Figure 7 shows the flow chart for conversion of a BCD number into a binary number (HEX). The following example illustrates the process of converting a 5-digit BCD number into corresponding HEX number as shown below.

15431 = (1*10000)+(5*1000)+(4*100)+(3*10)+(1*1)

=	1*2710H	=	2710H
	5*03E8H	=	1388H
	4*0064H	=	0190H
	3*000AH	=	001EH
+	1*0001H	=	0001H
	15431D	=	3C47H

First read all 5 BCD digits (from most significant digit to least significant digit) from five consecutive memory locations and store them in r0 to r4 respectively. Read the most significant digit from r0 and multiply it with 2710H, then store the result in the registers r6, r5. Then read next digit from r1 and multiply it with 03E8H and add this result to the previous result, and result of addition will be stored back into the same registers r6, r5. Similarly the consecutive digits are read and are multiplied by numbers 0064H, 000AH respectively and after each multiplication it will be added to the previous result. Finally, the last nibble is added to the result. After final addition a 16-bit binary number will be available in registers r6 (high byte) and r5 (low byte).

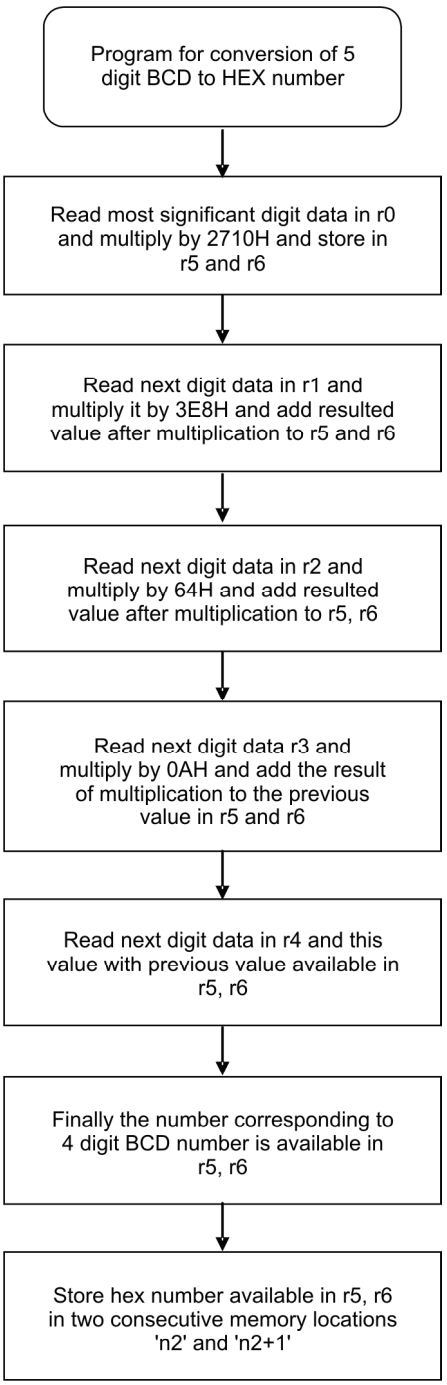

Program for conversion of 5 digit BCD to HEX number
Read most significant digit data in r0 and multiply by 2710H and store in r5 and r6
Read next digit data in r1 and multiply it by 3E8H and add resulted value after multiplication to r5 and r6
Read next digit data in r2 and multiply by 64H and add resulted value after multiplication to r5, r6
Read next digit data r3 and multiply by 0AH and add the result of multiplication to the previous value in r5 and r6
Read next digit data in r4 and this value with previous value available in r5, r6
Finally the number corresponding to 4 digit BCD number is available in r5, r6
Store hex number available in r5, r6 in two consecutive memory locations 'n2' and 'n2+1'

Fig. 7. Flow chart for conversion of 5-digit BCD number into HEX number.

Procedure

1. Load the unpacked 5 digit BCD number (number <=65535) in five consecutive memory locations starting from the address 9100h to 9104h.
2. Load the program-7 into the program memory from the stating address 9000h.
3. Execute the program and verify the result in two consecutive memory locations 9105h (high byte) and 9106h (low byte).

Program 7 : Program for converting 5-digit BCD bit number into HEX number

Address	Opcode	Label	Mnemonic		Operand	Comments
			org	9100h		
9100	06 05 05 03 05	n1:	db	06h,05h,05h,03h,05h		;5 digit BCD number
9105	00 00	n2:	db	00h,00h		;to store HEX number
			org	9000h		
9000	90 91 00		mov		dptr,#n1	;load DPTR with address n1
9003	E0		movx		a,@dptr	;read high digit of n1
9004	F8		mov		r0,a	;store it in r0
9005	A3		inc		dptr	;increment address pointer by one
9006	E0		movx		a,@dptr	;read next digit of n1 store high
9007	F9		mov		r1,a	;store it in r1
9008	A3		inc		dptr	;increment address pointer by one
9009	E0		movx		a,@dptr	;read next digit of n1 store high
900A	FA		mov		r2,a	;store it in r2
900B	A3		inc		dptr	;increment address pointer by one
900C	E0		movx		a,@dptr	;read next digit of n1 store high
900D	FB		mov		r3,a	;store it in r3
900E	A3		inc		dptr	;increment address pointer by one
900F	E0		movx		a,@dptr	;read last digit of n1 store it in r4
9010	FC		mov		r4,a	;multiplication of high digit with ;2710h = 10000
9011	E8		mov		a,r0	;read high digit into acc.
9012	75 F0 10		mov		b,#10h	;load b with 10h
9015	A4		mul		ab	;multiply with high digit
9016	AE F0		mov		r6,b	;store high byte of product into r6
9018	FD		mov		r5,a	;store low byte of product into r5
9019	E8		mov		a,r0	;read high digit into acc.
901A	75 F0 27		mov		b,#27h	;store b with 27h
901D	A4		mul		ab	;multiply with high digit
901E	2E		add		a,r6	;add low byte of product with ;high byte of previous product
901F	FE		mov		r6,a	;store it in r6 ;multiplication of next digit with 03e8h = 1000
9020	E9		mov		a,r1	;read next digit into acc.
9021	75 F0 E8		mov		b,#0e8h	;load b with e8h
9024	A4		mul		ab	;multiply with high digit
9025	2D		add		a,r5	;add low byte of product with ;low byte of previous result
9026	FD		mov		r5,a	;store it in r5

Program 7 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
9027	E5 F0		mov	a,b	;read high byte of product into acc.
9029	3E		addc	a,r6	;add high byte of product with ;high byte of previous product along with ;carry
902A	FE		mov	r6,a	;store it in r6
902B	E9		mov	a,r1	;read same digit into acc.
902C	75 F0 03		mov	b,#03h	;load b with 03
902F	A4		mul	ab	;multiply with the digit
9030	2E		add	a,r6	;add high byte of product with ;high byte of previous result
9031	FE		mov	r6,a	;store it in r6 ;multiplication of next digit with 64h = 100
9032	EA		mov	a,r2	;read next digit into acc.
9033	75 F0 64		mov	b,#64h	;load b with 64h
9036	A4		mul	ab	;multiply with the digit
9037	2D		add	a,r5	;add low byte of product with low ;byte of previous result
9038	FD		mov	r5,a	;store it in r5
9039	E5 F0		mov	a,b	;read high byte of product into acc.
903B	3E		addc	a,r6	;add high byte of product with ;high byte of previous result
903C	FE		mov	r6,a	;store it in r6 ;multiplication of next digit with 0ah = 10
903D	EB		mov	a,r3	;read next digit into acc.
903E	75 F0 0A		mov	b,#0ah	;load b with 0ah
9041	A4		mul	ab	;multiply with the digit
9042	2D		add	a,r5	;add low byte of product with ;low byte of previous result
9043	FD		mov	r5,a	;store it in r5
9044	E5 F0		mov	a,b	;read high byte of product into acc.
9046	3E		addc	a,r6	;add high byte of product with ;high byte of previous result
9047	FE		mov	r6,a	;store it in r6 ;add next digit directly with the result
9048	EC		mov	a,r4	;read next digit into acc.
9049	2D		add	a,r5	;add digit with low byte of result
904A	FD		mov	r5,a	;store it in r5
904B	74 00		mov	a,#00	;clear accumulator
904D	3E		addc	a,r6	;add acc. with the high byte of ;result along with carry
904E	FE		mov	r6,a	;store it in r6 ;store result in two consecutive memory locations
904F	90 91 05		mov	dptr,#n2	;load DPTR with address of 'n2'
9052	F0		movx	@dptr,a	;store high byte of the result

Program 7 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
9053	ED		mov	a,r5	;read low byte of the result from r5
9054	A3		inc	dptr	;increment address pointer
9055	F0		movx	@dptr,a	;store it the 'n2+1' location
9056	80 FE		sjmp	$	;stay in infinite loop
			end		;end of the program

Experiment 8

AIM: Assembly language program to convert HEX number into corresponding BCD number.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A.

Working Procedure of Program: Figure 8 shows the flow chart for conversion of binary (HEX) number into BCD form. In a binary number, each bit position represents a power of 2. An 8-bit binary number, for example, can be represented as

$$b7*2^7 + b6*2^6 + b5*2^5 + b4*2^4 + b3*2^3 + b2*2^2 + b1*2^1 + b0*2^0$$

this can be shifted around and expressed as binary number as shown below

$$= ((((((2*b7+b6)*2+b5)*2+b4)*2+b3)*2+b2)*2+b1)*2+b0$$

where, b7 to b0 are the values of the binary bits. If we start with a binary number and do each operation in the nested parentheses in BCD with the aid of the DAA instruction, the result will be the BCD number equivalent to the original binary number.

A 16-bit binary number is applied as input to this program. This program produces two BCD digits (packed BCD number) of the result at a time by calling sub-program 'hex_bcd'. The main principle here is to shift the 16-bit binary number to the left by one bit position, so the MSB goes into the carry flag, this bit is added twice to the previous result. The DAA instruction is used to keep the result of the addition in BCD format, if the DAA produces a carry it will be added back into the shifted 16-bit number in registers r0 & r1 so that it will be propagated into higher digits. After each run of 'hex_bcd' (16 times of BCD_2), r0 & r1 will be left with a binary number minus the value of the two BCD digits produced. This procedure finally results in the generation of a 5-digit packed BCD number.

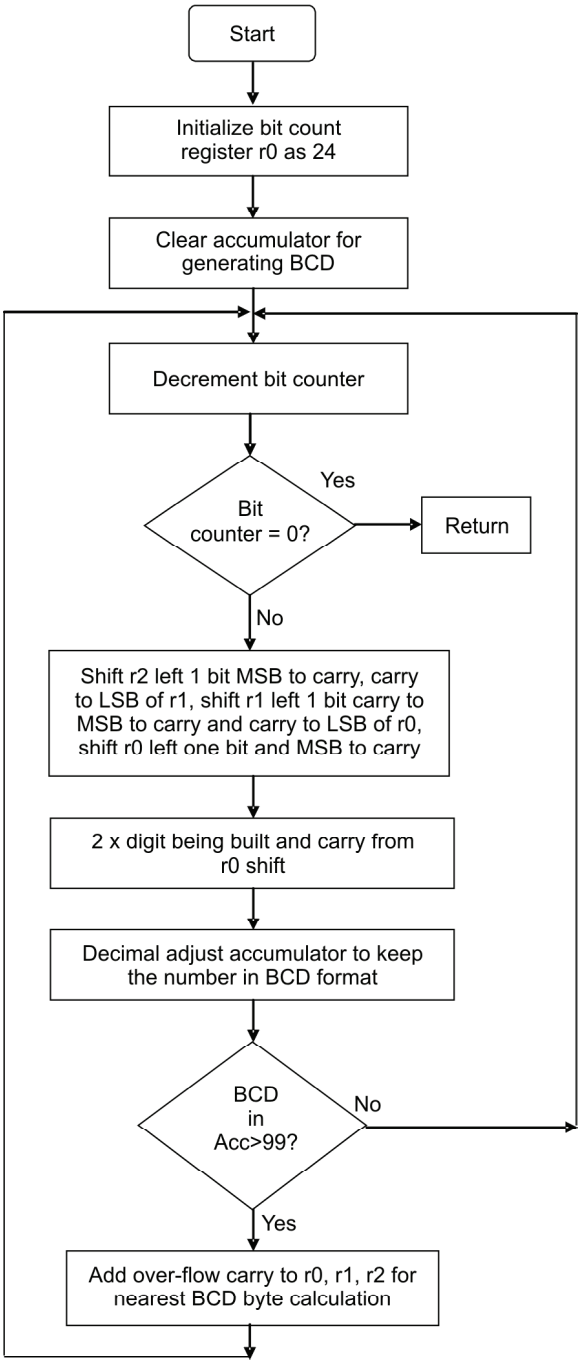

Start
Initialize bit count register r0 as 24
Clear accumulator for generating BCD
Decrement bit counter
Bit counter = 0?
Yes
Return
No
Shift r2 left 1 bit MSB to carry, carry to LSB of r1, shift r1 left 1 bit carry to MSB to carry and carry to LSB of r0, shift r0 left one bit and MSB to carry
2 x digit being built and carry from r0 shift
Decimal adjust accumulator to keep the number in BCD format
BCD in Acc>99?
No
Yes
Add over-flow carry to r0, r1, r2 for nearest BCD byte calculation

Fig. 8. Flow chart for subprogram to convert 24-bit binary number.

Procedure

1. Load the 16-bit HEX number in two consecutive memory locations starting from the address 9100h and 9101h.
2. Load the program-8 into the program memory from the starting address 9000h.
3. Execute the program and verify the result in three consecutive memory locations from 9102h to 9104h.

Program 8: Program for converting 16-bit HEX number into corresponding BCD number

Address	Opcode	Label	Mnemonic		Operand	Comments
			org	9100h		
9100	FF FF		hex:		db 0ffh,0ffh	;store HEX number
9102	00 00 00		bcd:		db 00h,00h,00h	;to store BCD number
			org	9000h		
9000	90 91 00		mov		dptr,#hex	;load DPTR with address 'hex'
9003	E0		movx		a,@dptr	;read high digit of HEX number
9004	F9		mov		r1,a	;store it in r1
9005	A3		inc		dptr	;increment address pointer by one
9006	E0		movx		a,@dptr	;read low byte of HEX number
9007	F8		mov		r0,a	;store it in r0
9008	7F 11		mov		r7,#11h	;load r7 with count 17 for 16 bit ;conversion
900A	12 90 2A		lcall		hex_bcd	;call subroutine to convert into ;BCD digits
900D	EB		mov		a,r3	;read 2 LS BCD digits from r3
900E	FC		mov		r4,a	;store them in r4
900F	7F 11		mov		r7,#11h	;load r7 with count 17
9011	12 90 2A		lcall		hex_bcd	;call conversion subroutine
9014	EB		mov		a,r3	;read next 2 BCD digits from r3
9015	FD		mov		r5,a	;store them in r5
9016	7F 11		mov		r7,#11h	;load r7 with count 17
9018	12 90 2A		lcall		hex_bcd	;call conversion subroutine
901B	EB		mov		a,r3	;read next 2 BCD digits from r3
901C	FE		mov		r6,a	;store them in r6
901D	90 91 02		mov		dptr,#result	;load DPTR with address of 'bcd'
9020	EE		mov		a,r6	;read high byte of BCD num. from r6
9021	F0		movx		@dptr,a	;store high byte of the result
9022	A3		inc		dptr	;increment address pointer
9023	ED		mov		a,r5	;read middle byte of BCD from r5
9024	F0		movx		@dptr,a	;store high byte of the result in ;'bcd+1'
9025	A3		inc		dptr	;increment address pointer
9026	EC		mov		a,r4	;read last byte of BCD from r5
9027	F0		movx		@dptr,a	;store high byte of the result in ;'bcd+2'
9028	80 24		sjmp		end_p	;go to end of the program

Program 8 : Contd.

Address	Opcode	Label	Mnemonic		Operand	Comments
902A	C3		hex_bcd:	clr	c	;clear carry flag
902B	74 00		mov		a,#00h	;clear accumulator
902D	FB		mov		r3,a	;store acc. into r3
902E	C3		bcd_2:	clr	c	;clear carry flag
902F	74 00		mov		a,#00h	;clear accumulator
9031	DF 01		djnz		r7, rep	;decrement r7 by one jump to ;location 'rep' for shifting and ;adding the digit with same
9033	22		ret			;return to main program
9034			rep:			
9034	E8		mov		a,r0	;store r0 contents in acc.
9035	33		rlc		a	;rotate acc. right through carry, ;MSB to carry
9036	F8		mov		r0,a	;store acc. back to r0
9037	E9		mov		a,r1	;store r1 contents in acc.
9038	33		rlc		a	;rotate acc. right through carry, ;carry to LSB of r1
9039	F9		mov		r1,a	;store acc. back to r1
903A	EB		mov		a,r3	;read r3 contents
903B	F5 F0		mov		b,a	;store them in b
903D	35 F0		addc		a,b	;add acc along with carry ;shifted from r1
903F	D4		da		a	;keep result in BCD
9040	FB		mov		r3,a	;store BCD number back in r3
9041	50 EB		jnc		bcd_2	;no carry from DAA, continue the ;same procedure if carry, ;propagate to r1 and r0 for ;future terms
9043	E8		mov		a,r0	;read r0 into acc.
9044	34 00		addc		a,#00h	;add carry to acc.
9046	F8		mov		r0,a	;store acc. back to r0
9047	E9		mov		a,r1	;read r0 into acc.
9048	34 00		addc		a,#00h	;add carry to acc.
904A	F9		mov		r1,a	;store acc. back to r1
904B	02 90 2E		ljmp		bcd_2	;repeat the conversion
904E	80 FE		end_p:	sjmp	$	;stay in infinite loop
				end		;end of the program

Experiment 9

AIM: Assembly language program for arranging the given data in ascending order.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A.

Working Procedure of Program: Initialize outer and inner counters r0 and r1 with n-1 count (where n is the number of data bytes present in the array). Load DPTR with the starting address of the array. Read the data byte of array and store it in the register B. Increment the address in DPTR to point next data byte and read the data byte. Compare it with the previous byte stored in the register B. If A=B then repeat the above steps, else exchange the contents of A and B. Store register A contents in the current memory location and store register B contents in the previous location by decrementing DPTR. Again increment DPTR to compare next two consecutive memory locations. Repeat the above procedure until r1 becomes 00h. When r1 becomes 00h, then decrement r0 by one. If r0 is not equal to 00h then re-initialize DPTR with starting address of array and again load r0 with n-1 count and repeat the above steps. When both registers r1 and r0 equal to 00h then stay in infinite loop.

Procedure

1. Load the data in the consecutive memory locations starting from address 9100h to 9109h.
2. Load the program-9 into the program memory from the starting address 9000h.
3. Execute the program and verify the given data in the memory locations 9100h to 9109h. The data bytes will be arranged in the ascending order.

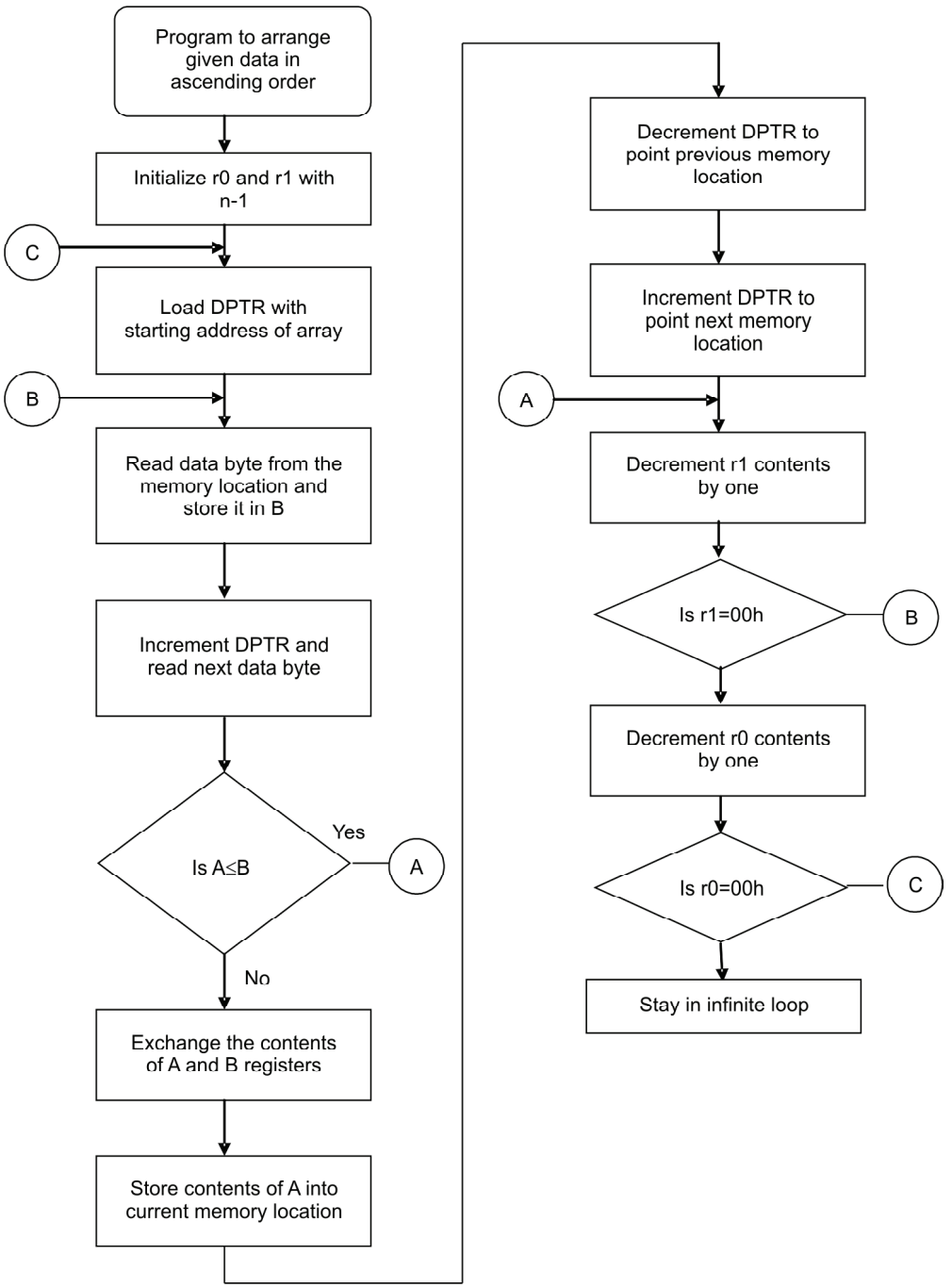

Program to arrange given data in ascending order
Initialize r0 and r1 with n-1
C
Load DPTR with starting address of array
B
Read data byte from the memory location and store it in B
Increment DPTR and read next data byte
Is A≤B
Yes
A
No
Exchange the contents of A and B registers
Store contents of A into current memory location
Decrement DPTR to point previous memory location
Increment DPTR to point next memory location
A
Decrement r1 contents by one
Is r1=00h
B
Decrement r0 contents by one
Is r0=00h
C
Stay in infinite loop

Fig. 9. Flow chart for arranging the given data in ascending order.

Program 9 : Program for arranging the given data in ascending order

Address	Opcode	Label	Mnemonic	Operand	Comments
			org 9100h		;data starts here
9100 9105	69 20 40 78 01 50 02 05 08 00		array: db	69h,20h,40h,78h,01h,50h,02h,05h,08h,00h	
			org 9000h		
9000	78 09		mov	r0,#09h	;Initialize r0 with n-1 ;where n is number of data bytes
9002	79 09	x4:	mov	r1,#09h	;Initialize r1 with n-1
9004	90 91 00		mov	dptr,#array	;Load DPTR with starting address ;of array
9007	E0	x3:	movx	a,@dptr	;read data byte of array
9008	F5 F0		mov	b,a	;store it in register B
900A	A3		inc	dptr	;increment DPTR
900B	E0		movx	a,@dptr	;read next data bye of array
900C	B5 F0 00		cjne	a,b,x1	;compare contents of register A ;with register B
900F	50 0A	x1:	jnc	x2	;jump if A>B, then repeat above ;steps
9011	C5 F0		xch	a,b	;else exchange A with B
9013	F0		movx	@dptr,a	;store A with current memory ;location
9014	15 82		dec	dpl	;decrement DPTR
9016	C5 F0		xch	a,b	;exchange A with B
9018	F0		movx	@dptr,a	;store with previous memory ;location
9019	05 82		inc	dpl	;increment DPTR
901B	D9 EA	x2:	djnz	r1,x3	;if r1 contents are 00H, then ;repeat above steps
901D	D8 E3		djnz	r0,x4	;else decrement r0, if r0!=00H, ;then reload r1 with 09h and ;repeat above steps
901F	80 FE		sjmp	$	;stay in infinite loop
			end		;end of the program

Experiment 10

AIM: Assembly language program for arranging the given data in descending order

EQUIPMENT: Microcontroller kit, power supply + 5V/3A.

Working Procedure of Program: Initialize outer and inner counters r0 and r1 with n-1 count (where n is the number of data bytes present in the array). Load DPTR with the starting address of the array. Read the data byte of array and store it in the register B. Increment the address in DPTR to point next data byte and read the data byte. Compare it with the previous byte stored in the register B. If A=B then repeat the above steps, else exchange the contents of A and B. Store register A contents in the current memory location and store register B contents in the previous location by decrementing DPTR. Again increment DPTR to compare next two consecutive memory locations. Repeat the above procedure until r1 becomes 00h. When r1 becomes 00h, then decrement r0 by one. If r0 is not equal to 00h then re-initialize DPTR with starting address of array and again load r0 with n-1 count and repeat the above steps. When both registers r1 and r0 equal to 00h then stay in infinite loop.

Procedure

1. Load the data in the consecutive memory locations starting from address 9100h to 9109h.
2. Load the program-10 into the program memory from the starting address 9000H.
3. Execute the program and verify the given data in the consecutive memory locations from 9000h to 9109h. The data bytes will be arranged in the descending order.

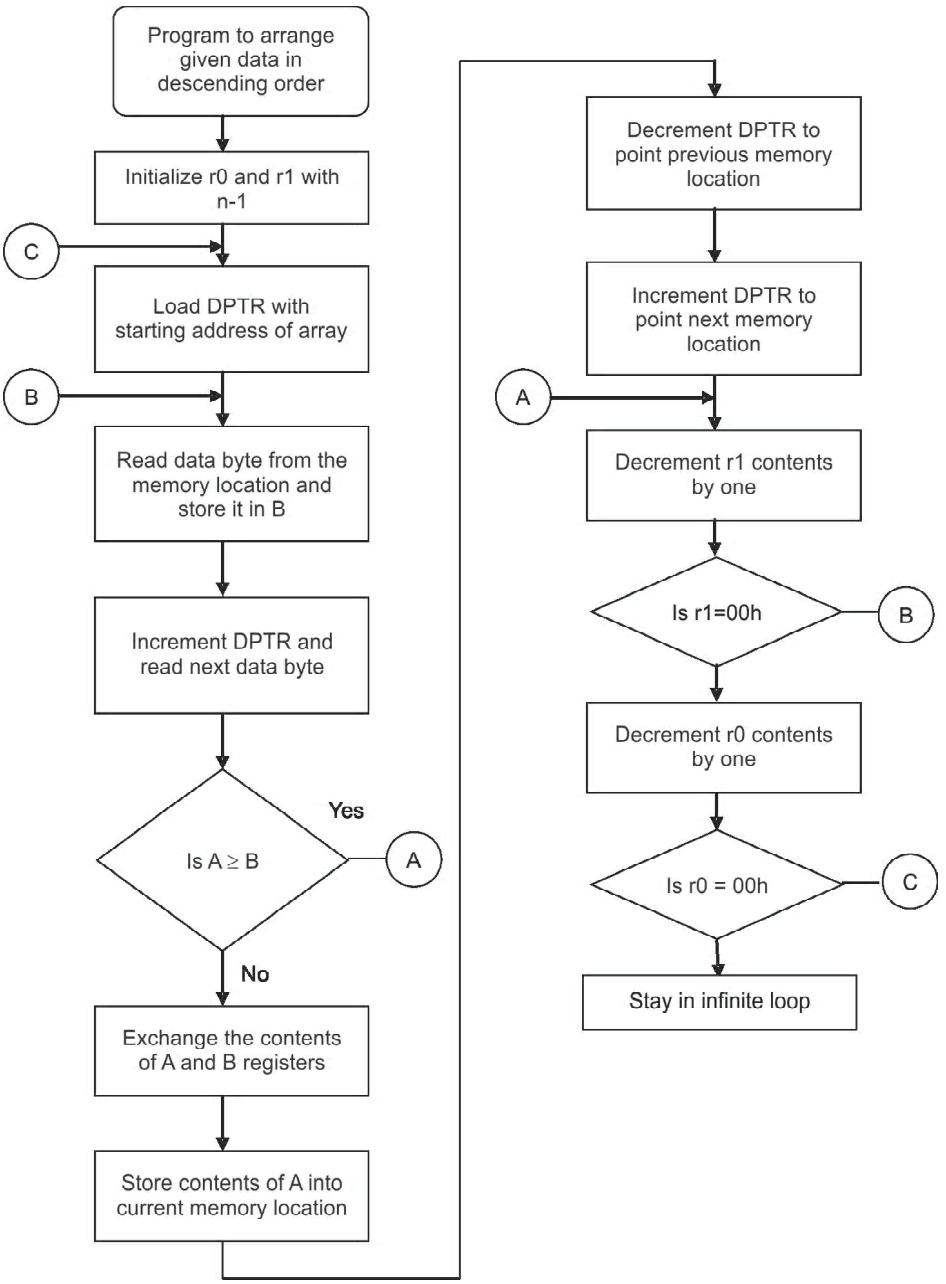

Program to arrange given data in descending order
Initialize r0 and r1 with n-1
C
Load DPTR with starting address of array
B
Read data byte from the memory location and store it in B
Increment DPTR and read next data byte
Is A ≥ B
Yes
A
No
Exchange the contents of A and B registers
Store contents of A into current memory location
Decrement DPTR to point previous memory location
Increment DPTR to point next memory location
A
Decrement r1 contents by one
Is r1=00h
B
Decrement r0 contents by one
Is r0 = 00h
C
Stay in infinite loop

Fig. 10. Flow chart for arranging the given data in descending order.

Program 10 : Program for arranging the given data in descending order

Address	Opcode	Label	Mnemonic		Operand	Comments
			org	9100h		;data starts here
9100 9105	69 20 40 78 01 50 02 05 08 00			array: db	69h,20h,40h,78h,01h,50h,02h,05h,08h,00h	
			org	9000h		
9000	78 09		mov		r0,#09h	;Initialize r0 with n-1 ;where n is number of data bytes
9002	79 09	x4:	mov		r1,#09h	;Initialize r1 with n-1
9004	90 91 00		mov		dptr,#array	;Load DPTR with starting address ;of array
9007	E0	x3:	movx		a,@dptr	;read data byte of array
9008	F5 F0		mov		b,a	;store it in register B
900A	A3		inc		dptr	;increment DPTR
900B	E0		movx		a,@dptr	;read next data bye of array
900C	B5 F0 00		cjne		a,b,x1	;compare contents of register A ;with register B
900F	50 0A	x1:	jc		x2	;jump if A=B, then repeat above ;steps
9011	C5 F0		xch		a,b	;else exchange A with B
9013	F0		movx		@dptr,a	;store A with current memory location
9014	15 82		dec		dpl	;decrement DPTR
9016	C5 F0		xch		a,b	;exchange A with B
9018	F0		movx		@dptr,a	;store with previous memory location
9019	05 82		inc		dpl	;increment DPTR
901B	D9 EA	x2:	djnz		r1,x3	;if r1 contents are 00H, then ;repeat above steps
901D	D8 E3		djnz		r0,x4	;else decrement r0, if r0!=00h, ;then reload r1 with 09h and ;repeat above steps
901F	80 FE		sjmp		$	;stay in infinite loop
			end			;end of the program

PART II
Hardware Experiments

Experiments 11

AIM: To interface a binary counter display with microcontroller.

COMPONENTS: IC 74LS573, 8-LEDs, 8-330Ω resistors, bread board and connecting wires.

EQUIPMENT: Microcontroller kit, power Supply + 5V/3A.

INTRODUCTION: A binary counter is like any other counter which counts the numbers in binary form. Here, an 8-bit binary counter is designed using 8-LEDs. This will indicate the binary sequence from 00000000 to 11111111. A binary '1' will glow the LED and binary '0' will off the LED. The port-1 of microcontroller is used to generate an 8-bit binary sequence. To glow a LED, it requires a minimum 5mA of current, but microcontroller port cannot source this current, hence, a buffer must be interfaced between LED and the microcontroller.

Hardware: Figure 11 shows the interfacing of binary counter with microcontroller. In the present experiment 8-LEDs are interfaced to the microcontroller port-1 through the IC 74LS573. The IC 74LS573 acts as a buffer, which provides enough current to drive the LEDs.

Software: Figure 12 shows the flow chart of binary counter program. Initially the program clears port-1 to switch off all LEDs. A software delay has been incorporated between two successive counts so that an experimenter can visualize the binary counting sequence. After delay the port-1 count is incremented by one. In this way the software increments the count from 00000000 to 11111111 in steps of 1. This sequence can be observed by the glow of LEDs.

The software delay is obtained with internal registers r0, r1, and r2. Initially the registers r0, r1, and r2 are loaded with the count 08H, FFH, FFH respectively. The microcontroller decrements the contents of r2 register by one and checks whether the count has become zero or not, if the count is not zero then again it decrements the contents by one. This process is repeated until r2 register contents becomes zero. If count becomes zero, then it decrements r1 count by one. Again register r2 is loaded with count FFH. The above procedure is repeated until contents of r1 become 00H. If r1 register contents become zero, then it decrements r0 register contents by one and registers r1 & r2 are loaded again with FFH, FFH. The above procedure is

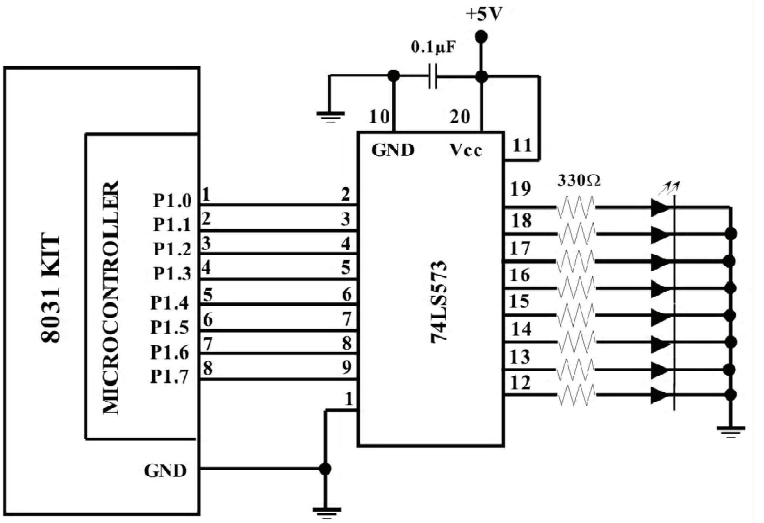


Fig. 11. Interfacing of binary counter with microcontroller.

repeated until r0 contents become 00H. The microcontroller will be in the delay subroutine until the registers r0,r1 and r2 contents become 00H. When count becomes zero then program control returns to the main program.

Procedure

1. Connect the circuit as shown in Fig. 11.
2. Load the assembly language Program-11 (or embedded 'C' program) into the program memory from the starting address 8000H of the microcontroller kit.
3. Execute the program. Observe the glow of LEDs in the binary sequence.

Program 11 : Program for binary counter

Address	Opcode	Label	Mnemonic	Operand	Comments
8000	C2 81		clr	p1	;clear port-1
8002	12 80 0A	dis:		lcall delay	;call delay subroutine
8005	05 90		inc	p1	;increment port-1 contents by ;one
8007	02 80 02		ljmp	dis	;go to the label dis
800A	78 08	delay:	mov	r0,#08h	;delay of about 1sec
800C	79 FF	l3:	mov	r1,#0ffh	
800E	7A FF	l2:	mov	r2,#0ffh	
8010	DA FE	l1:	djnz	r2,l1	
8012	D9 FA		djnz	r1,l2	
8014	D8 F6		djnz	r0,l3	
8016	22		ret		;return to main program
8017		end			;program ends

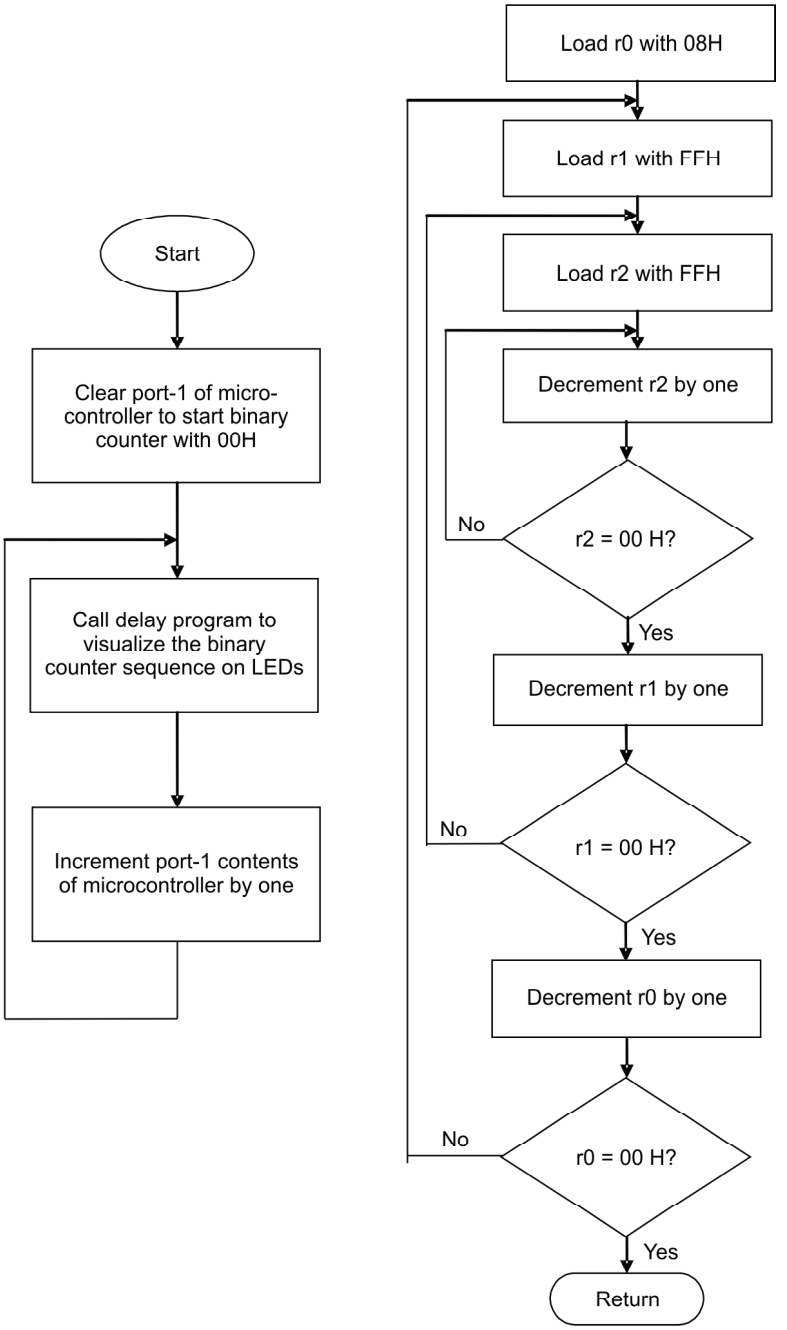


Fig. 12. Flow chart for binary counter and delay program.

Experiment 12

AIM: To interface a seven-segment display (SSD) with microcontroller.

COMPONENTS: IC74LS573, seven-segment display (common cathode), 8-330Ω resistors, bread board, and connecting wires.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A.

INTRODUCTION: The seven-segment display (SSD) is a multiple display used for displaying decimal digits, some letters and characters. Though there are many other multiple displays like 9-segment, 14-segment and 18-segment, SSD is widely used because of its simple circuit requirements. In SSD, seven segments (LEDs) are arranged in the form of decimal digit 8, and different digits and characters can be displayed by glowing the corresponding segments. There are two types of SSDs available viz., common-cathode and common-anode type SSDs. In a common-cathode type of SSD, the cathodes of all segments are internally tied together and this is to be connected to the ground. The desired segments are glown by applying a logic '1' (+5V) to the anodes of the corresponding segments through the current limiting resistors. In common-anode type, anodes of all segments are internally tied together and this is to be connected to +5V supply. Applying logic '0' (0V) to the cathode, the corresponding segment is made to glow.

Hardware: Figure 13 shows the interfacing of a common-cathode type seven-segment display with microcontroller. Since the microcontroller cannot source the current required to glow SSD, hence it is driven through the buffer/latch IC 74LS573.

Software: Figure 14 shows the flow chart of the program to display the numbers from 0 to F on SSD. Initialize the register r7 with 10H, which acts as a counter to count the numbers 0-F to be displayed on SSD. Load DPTR with the starting address of the memory location, where the seven-segment codes for the numbers 0 to F are stored. Read the first code from the memory and send to port-1 of microcontroller to display it on SSD. Call a delay subroutine to visualize the number displayed on

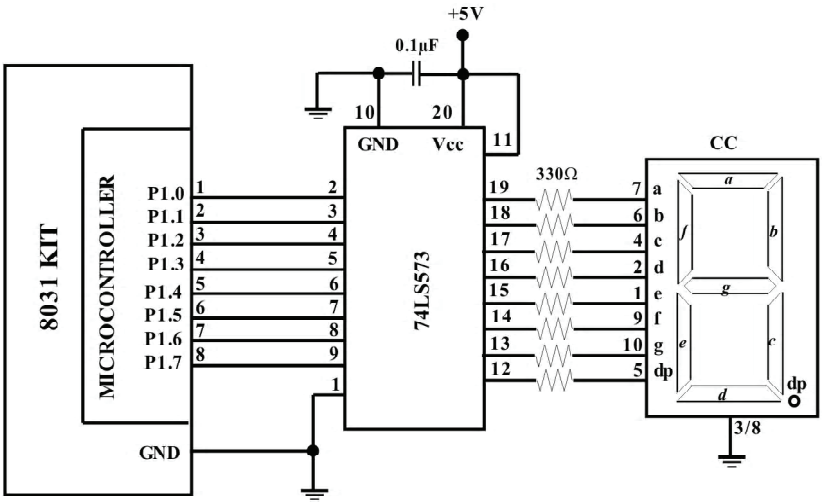


Fig. 13. Flow chart for binary counter and delay program.

the SSD. After the delay, decrement the content of r7 by one. If the content of r7 is not zero, then increment data pointer by one to fetch the next seven-segment code. Read it and send to the SSD. This procedure is repeated until the content of r7 becomes zero. If r7 contents are zero, then execution goes to the beginning of the program to display the numbers from 0 – F again.

Procedure

1. Wire the circuit as shown in the Fig. 13.
2. Load the assembly language program-12 (or embedded 'C' program) into the program memory from the starting address 8000H. And load the seven-segment codes from the starting address 8100H of microcontroller kit.
3. Execute the program. Observe the display of numbers from 0 to F on seven-segment display continuously.

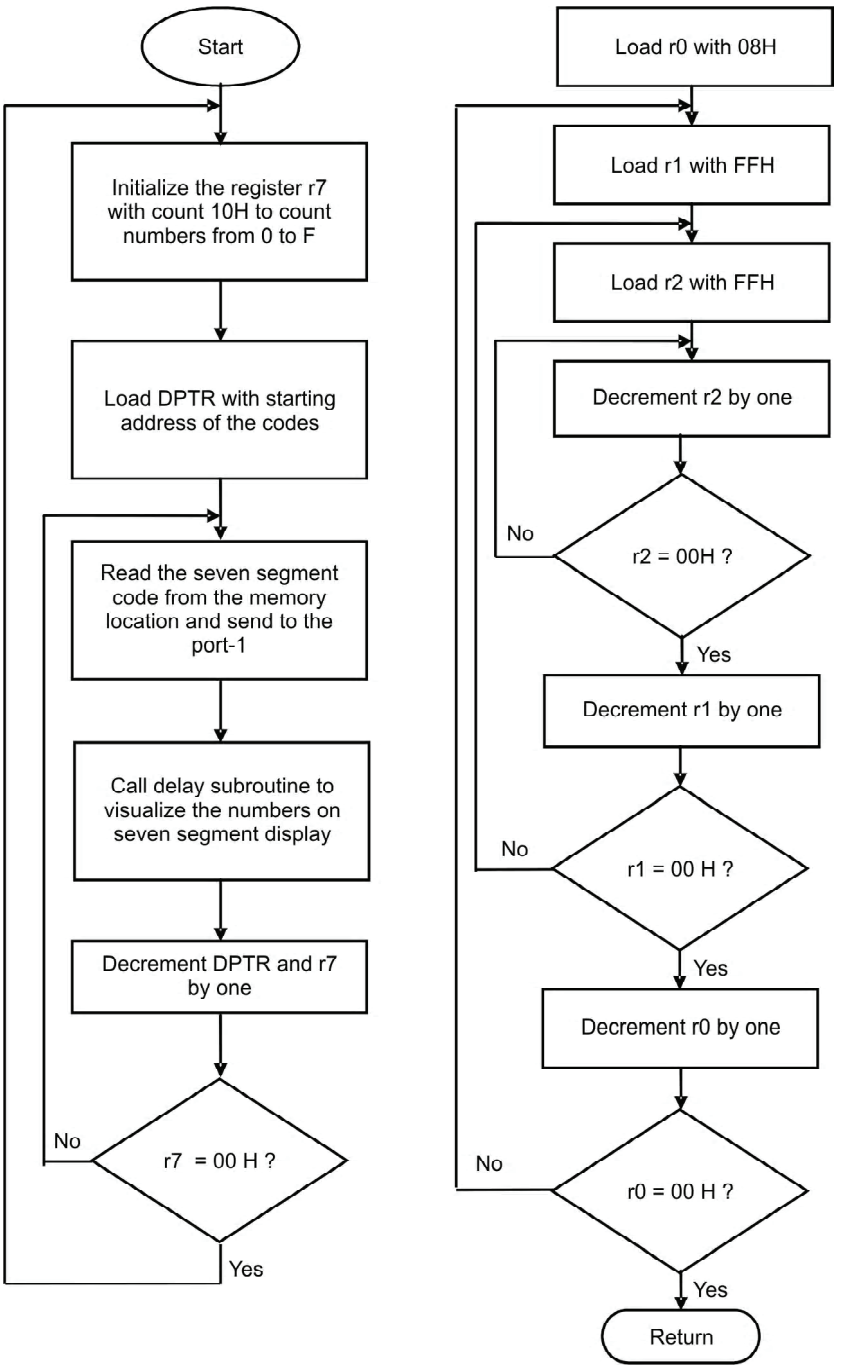

Start
Initialize the register r7 with count 10H to count numbers from 0 to F
Load DPTR with starting address of the codes
Read the seven segment code from the memory location and send to the port-1
Call delay subroutine to visualize the numbers on seven segment display
Decrement DPTR and r7 by one
r7 = 00 H ?
No
Yes
Load r0 with 08H
Load r1 with FFH
Load r2 with FFH
Decrement r2 by one
r2 = 00H ?
No
Yes
Decrement r1 by one
r1 = 00 H ?
No
Yes
Decrement r0 by one
r0 = 00 H ?
No
Yes
Return

Fig. 14. Interfacing of seven-segment display with microcontroller.

Program 12 : Program for displaying 0-F on seven-segment display

Address	Opcode	Label	Mnemonic	Operand	Comments
					;Common cathode seven segment codes
8100	3F 06 5B 4F 66	sev_seg:	db	3fh,06h,5bh,4fh,66h,6dh,7dh,07h	
8105	6D 7D 07				
8108	7F 6F 77 7C 39		db	7fh,6fh,77h,7ch,39h,5eh,79h,71h	
810D	5E 79 71				
					;program starts here
8000	7F 10	dis:	mov	r7,#10h	; Initialize the register r0 with ;with count 10H
8002	90 81 00	mov		dptr,#sev_seg	;Load data pointer (DPTR) ;with starting address of codes
8005	E0	get_code:	movx	a,@dptr	;read seven segment code
8006	F5 90	mov		p1,a	
8008	12 80 11	lcall		delay	;call delay subroutine
800B	A3	inc		dptr	;increment address in DPTR
800C	DF F7	djnz		r7,get_code	;decrement contents of r7, if ;contents ?0, then go to label ;get_c
800E	02 80 00	ljmp		dis	;go to label dis
					;delay subroutine
8011	78 08	delay:	mov	r0,#08h	;delay of about 1 sec
8013	79 FF	l3:	mov	r1,#0ffh	
8015	7A FF	l2:	mov	r2,#0ffh	
8017	DA FE	l1:	djnz	r2,l1	
8019	D9 FA		djnz	r1,l2	
801B	D8 F6		djnz	r0,l3	
801D	22		ret		;return to main program
801D	end				;program ends

Experiment 13

AIM: To interface a multiplexed display with microcontroller.

COMPONENTS: BCD to seven-segment decoder/ driver 74LS47, 4-common anode seven segment displays, 4-2N2222 NPN transistors, 8-330 Ω, 4-1k Ω resistors, bread board and connecting wires.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A.

INTRODUCTION: In a static display the current through the display is being passed at all times. Hence, the static display consumes more power, this feature is not desirable for battery operated instruments. Moreover, in the static display each seven-segment display requires a separate seven-segment decoder/ driver and a port. Hence, the hardware requirement is more in static display. However, the multiplexed display overcomes these problems.

Hardware: Figure 15 shows the schematic diagram of four-digit multiplexed display. Here, 4-common anode seven-segment displays are interfaced to the microcontroller through BCD to seven-segment decoder/driver (74LS47). In this circuit only one 74LS47 decoder is used and its outputs are connected to the inputs of all displays. The microcontroller sends four-bit BCD data to the 74LS47, the decoder decodes this BCD data into seven-segment format and sends to all displays on a common bus, but only one display is turned ON at a time. The NPN transistor in series with the common-anode of each display acts as a switch to turn ON/OFF power supply to the display.

The BCD code for digit-1 is first output from port-1 (P1.0 – P1.3) to the 74LS47. The 74LS47 outputs the corresponding seven-segment code on the segment bus lines, then the transistor connected to digit-1 is turned ON by outputting a HIGH on P1.7. The rest of the bits of port-1 (P1.6 – P1.4) will be LOW to make sure that no other digits are turned ON. Hence, the number is displayed on digit-1. After 2 msec all the digits are turned OFF by outputting LOWs to port-1. The BCD code for digit-2 is then output to the 74LS47 on port-1 (P1.0 – P1.3) and HIGH on P1.6 to turn ON

digit-2. Hence, the number is displayed on digit-2. Then after 2 msec again all displays are turned OFF. This process is repeated for digit-3 and digit-4. After glowing all four digits, the digit-1 and the following digits are lit again. With 4-digits and 2 msec per digit, microcontroller gets back to digit-1 after every 8 msec or 80 times per second. This refresh rate is fast enough for our eyes that the digits will appear to glow all at a time. Though the multiplexed display reduces hardware and power consumption, there is only disadvantage in software multiplexed display approach is that it puts an additional burden on microcontroller.

Software Description

a. Clear accumulator
b. Send contents of accumulator to port-1 to clear the display
c. Load register r0 with count 04H
d. Load DPTR with starting address of the numbers to be displayed
e. Read number from the memory through DPTR
f. Send to decoder to display on particular seven segment display
g. Call delay (~2msec) subroutine
h. Clear display
i. Increment DPTR to point to next memory location to read next number to be displayed
j. Decrement r0. If r0≠00H, then repeat above steps from (e)
k. If r0=00H,then repeat steps from (a)

Procedure

1. Wire the circuit as shown in Fig. 15.
2. Load the assembly language program -13 (or embedded 'C' program) into the program memory from the starting address 8000H and store the numbers to be displayed from the starting address 8100H of microcontroller kit.
3. Execute the program and observe the continuous display of numbers '1234' on four seven-segment displays.

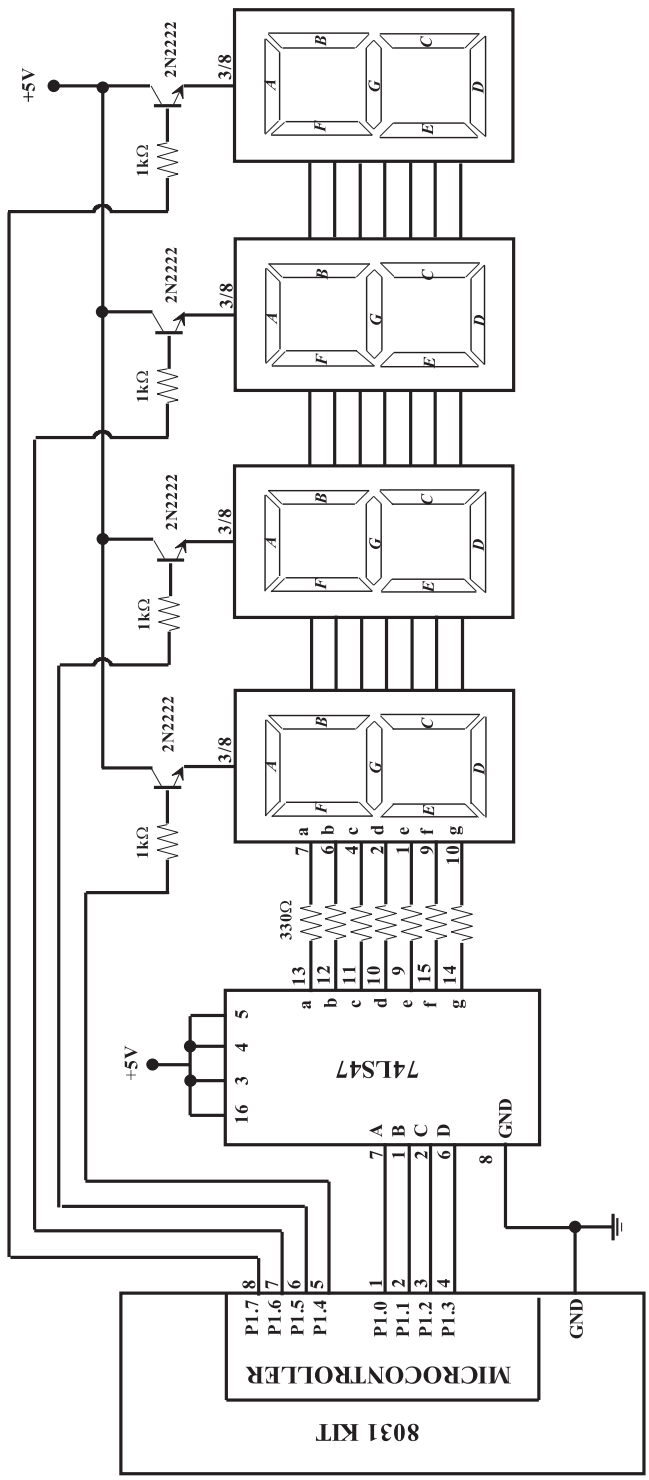


Fig. 15. The schematic diagram of four-digit multiplexed display.

Program 13 : Program for four-digit multiplexed display

Address	Opcode	Label	Mnemonic	Operand	Comments
					;store the numbers (1,2,3,4) to be displayed
		org	8100h		
8100	81 42 23 14	dat_dis:	db	81h,42h,23h,14h	
		org	8000h		
8000	75 81 40	mov		sp,#40h	;initialize stack pointer ;with starting address 40h
8003	E4	x2:	clr	a	;clear accumulator
8004	F5 90	mov		p1,a	;clear port-1
8006	78 04	mov		r0,#04h	;load r0 with count 04
8008	90 81 00	mov		dptr,#dat_dis	;load DPTR with starting ;address of the data
800B	E0	x1:	movx	a,@dptr	;read data
800C	F5 90	mov		p1,a	;send to display
800E	12 80 18	lcall		delay	;call delay subroutine
8011	E4	clr		a	;clear accumulator
8012	F5 90	mov		p1,a	;send to 00h to blank ;display
8014	D8 F5	djnz		r0,x1	;decrement r0 by one ;if r0!=00h go to x1 location
8016	80 EB	sjmp		x2	;enter into continuous loop ;delay subroutine
8018	79 05	delay:	mov	r1,#5	
801A	7A FF	x3:	mov	r2,#0ffh	
801C	DA FE	djnz		r2,$	
801E	D9 FA	djnz		r1,x3	
8020	22	ret			
		end			

Experiment 14

AIM: 1. To interface a stepper motor with microcontroller.
2. To rotate the stepper motor in clockwise and counter clockwise directions with full step (1.8°) and half step (0.9°) angles.

COMPONENTS: Four-coil stepper motor (+12V/100mA), 74LS573, 4-2N2222 transistors, 4-SL100 transistors, 4-1KΩ resistors, 4-1N4001 diodes, bread board and connecting wires.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A and + 12V/500mA

INTRODUCTION: A stepper motor is different from an ac or dc motor. The ac or dc motors rotate continuously when the power is applied, while the stepper motor rotates in steps of precise angle in response to train of pulses. By suitable control of windings, a continuous rotation with speed control and finally stopping at any angular position can be obtained. The stepper motor got this name because every pulse of excitation to the windings steps the rotor by a precise angle called the step angle. It can rotate in either direction; start/stop and the speed can also be varied. The rate of the excitation pulses will decide the speed of the motor. There are three types of stepper motors:

(i) Variable reluctance type
(ii) Permanent magnet type and
(iii) Hybrid type

With the first two types we can obtain step angles of 90°, 30°, 22.5° and 15°only. To obtain fine step angles of the order of 1.8°, 0.9° we have to go for hybrid stepper motors.

Hardware: The schematic diagram of interfacing of hybrid stepper motor with the microcontroller is shown in Fig. 16. The excitation pulses to the stepper motor windings are given from port-1 (P1.0 to P1.3) of microcontroller. These signals are generally low power signals, so they can't drive the motor windings directly. Hence, a suitable

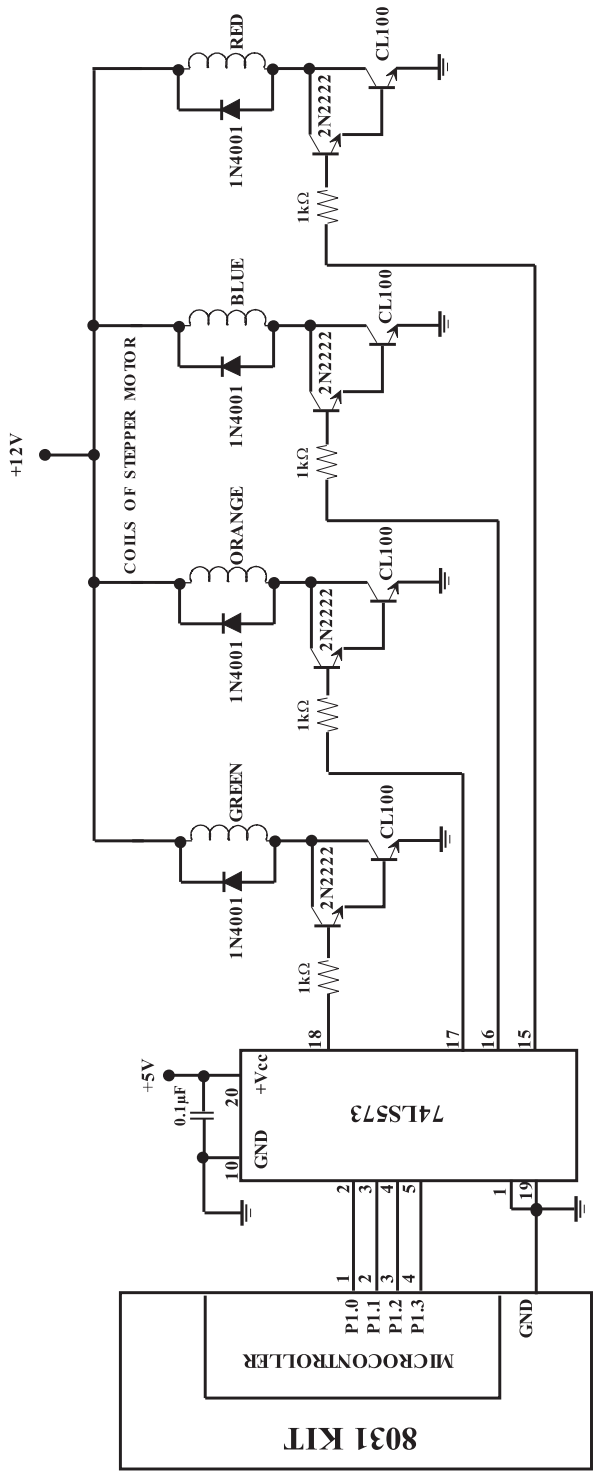


Fig. 16. The schematic diagram of interfacing of hybrid stepper motor with the microcontroller.

driving circuit has been employed. Here, the Darlington pairs are used in between microcontroller port lines and stepper motor coils as driving elements. In order to rotate the stepper motor with a step angle of 1.8° such as clockwise or anti-clockwise the switching sequence, given in the Table 1 must be sent to the bases of the Darlington pair transistors. To rotate the stepper motor with a step angle of 0.9° the switching sequence given in Table 2 must be sent to the bases of the Darlington pair transistors through microcontroller port lines.

Table 1

Step	TR4	TR3	TR2	TR1	CW	CCW
1.	0	0	1	1		
2.	1	0	0	1		
3.	1	1	0	0		
4.	0	1	1	0		

Table 2

Step	TR4	TR3	TR2	TR1	CW	CCW
1.	0	0	1	1		
2.	0	0	0	1		
3.	1	0	0	1		
4.	1	0	0	0		
5.	1	1	0	0		
6.	0	1	0	0		
7.	0	1	1	0		
8.	0	0	1	0		

Software: The codes that are to be sent to the stepper motor (in order to excite the appropriate winding) are stored in a look-up table from the starting address 8100H. The register r0 is used as a counter with initial value 04 i.e., the number of codes to be sent to the stepper motor windings. The codes are read from the look-up table into the accumulator and are sent to port-1. After sending the code, a delay subroutine is called, so that the stepper motor can respond to the next code. Every time after sending a code to port-1, the counter r0 is decremented by one. This process is repeated until counter r0 becomes zero. When counter contents becomes zero the counter r0 is again reloaded with count 04 and same procedure is repeated for the continuous rotation.

a. Initialize the data pointer (DPTR) to first code stored in 8100H
b. Initialize the counter (r0) with count 04 (for 1.8° step angle), 08 (for 0.9° step angle)
c. Read the contents of memory location pointed by DPTR into accumulator
d. Send the contents of accumulator to port-1
e. Call delay subroutine
f. Increment the contents of DPTR to point to next memory location
g. Decrement the counter (r0) contents by one

h. Compare the counter (r0) with 00H. If not equal repeat from step (c), if equal go to the next instruction.
i. Repeat the above steps from step (a) to rotate stepper motor continuously.

Experimental Procedure for 1.8° Step Angle

1. Wire the circuit as shown in Fig. 16.
2. Load the assembly language program-14 (or embedded 'C' program) into the program memory from the starting address 8000H in microcontroller kit, and store the numbers (03H, 09H, 0CH, 06H for clockwise direction and numbers 06H, 0CH, 09H, 03H for anti-clockwise direction) from the starting address 8100H to rotate the stepper motor with step angle of 1.8°.
3. Execute the program and observe the continuous rotation of the stepper motor. The stepper motor speed can be varied by changing the delay.

Experimental Procedure for 0.9° Step Angle

1. Wire the circuit as shown in Fig. 16.
2. Load the program-15 into the program memory from the starting address of microcontroller kit and store the numbers (03H, 01H, 09H, 08H, 0CH, 04H, 06H, 02H for clockwise direction and numbers 06H, 02H, 0CH, 04H, 09H, 08H, 03H for anti-clockwise direction) from the starting address 8100H to rotate the stepper motor with step angle of 0.9°.
3. Execute the program and observe the continuous rotation of the stepper motor. The stepper motor rotational speed can be varied by varying the delay.

Program 14 : Program for rotating stepper motor continuously with a 1.8° step angle

Address	Opcode	Label	Mnemonic	Operand	Comments
8100		org		8100h	
8100	03 09 0C 06	s_codes:	db	03h,09h,0ch,06h	;stepper motor codes ;for 1.8^0 step angle
8000		org		8000h	
8000	90 81 00	x2:	mov	dptr,#s_codes	;load DPTR with the ;starting of the codes
8003	78 04		mov	r0,#04h	;load r0 with count 04
8005	E0	x1:	movx	a,@dptr	;read code from memory
8006	F5 90		mov	p1,a	;send code to the stepper ;motor
8008	12 80 10	lcall		delay	;call delay subroutine
800B	A3	inc		dptr	;increment DPTR by one
800C	D8 F7	djnz		r0,x1	;decrement r0 by one ;If r0!=00, then go to ;location x1
800E	80 F0	sjmp		x2	;repeat steps from ;location x2 ;delay subroutine

Program 14 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
8010	79 FF	delay:	mov	r1,#0ffh	
8012	7A FF	x3:	mov	r2,#0ffh	
8014	DA FE	djnz		r2,$	
8016	D9 FA	djnz		r1,x3	
8018	22	ret			
8019		end			;program ends

Program 15 : Program for rotating stepper motor continuously with a 0.9° step angle

Address	Opcode	Label	Mnemonic	Operand	Comments
8100		org		8100h	
8100	03 01 09 08 0C 04 06 02	s_codes:	db	03h,01h,09h,08h 0ch,04h,06h,02h	;stepper motor codes ;for 0.9^0 step angle
8000		org		8000h	
8000	90 81 00	x2:	mov	dptr,#s_codes	;load DPTR with the ;starting of the codes
8003	78 04		mov	r0,#08	;load r0 with count 08
8005	E0	x1:	movx	a,@dptr	;read code from memory
8006	F5 90		mov	p1,a	;send code to the stepper ;motor
8008	12 80 10	lcall		delay	;call delay subroutine
800B	A3	inc		dptr	;increment DPTR by one
800C	D8 F7	djnz		r0,x1	;decrement r0 by one ;If r0!=00, then go to ;location x1
800E	80 F0	sjmp		x2	;repeat the steps from ;location x2 ;delay subroutine
8010	79 FF	delay:	mov	r1,#0ffh	
8012	7A FF	x3:	mov	r2,#0ffh	
8014	DA FE	djnz		r2,$	
8016	D9 FA	djnz		r1,x3	
8018	22	ret			
8019		end			;program ends

Experiment 15

AIM: 1. To interface an 8-bit D/A converter MC1408 with microcontroller
2. To convert digital data into corresponding analog voltage and
3. To generate waveforms, viz. square, rectangular, ramp, and triangular

COMPONENTS: D/A converter MC1408, Op-amp μA 741, 3-5.1kΩ resistors, bread board and connecting wires.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A, ±12V/100mA

INTRODUCTION: MC1408 is an inexpensive and versatile 8-bit digital to analog converter. It is compatible with TTL and CMOS logic. Hence, it is easy to interface with microprocessor/microcontroller. It has 8 input data lines. It requires 2 mA of reference current. The reference current is determined by the resistor R_{ref} and V_{ref}. The resistor R1 is generally equal to R_{ref} to match the input impedance of reference voltage source. The MC1408 is current output D/A converter.

Hardware: Interfacing of D/A converter with microcontroller is shown in Fig. 17. Port-1 of microcontroller is connected to the input data lines of D/A converter. A 5V (V_{ref}) supply and 5.1 kΩ (R_{ref}) gives a reference current of ≈ 1mA. The D/A converter output is current, hence, it is converted into voltage by current to voltage converter using operational amplifier as shown in the Fig. 17. Here, 5.1 kΩ is used as feedback resistor (R_f). The output current of D/A converter is calculated by the following formula.

$$I_o = V_{ref} / R_{ref} \left(\frac{d7}{2} + \frac{d6}{4} + \frac{d5}{8} + ... + \frac{d0}{256} \right)$$

The output of current to voltage converter is given by the following equation

$$Vo = Io * R_f$$

The output of the operational amplifier can be observed on a digital multimeter/ CRO.

Software Details

a. Load accumulator with digital data (from 00H to FFH)
b. Output the contents of the accumulator to the port-1
c. Measure the D/A converter output with multi-meter

Procedure

1. Wire the circuit as shown in Fig. 17.
2. Load the assembly language program-16 (or embedded 'C program) into the program memory from the starting address 8000H of microcontroller kit.
3. Execute the program. Observe the corresponding analog voltage at the output of D/A converter on digital multimeter.
4. Vary the digital data loaded into the accumulator from 00H – FFH in steps of 20H and observe the corresponding analog voltage at the output of D/A converter on digital multimeter.
5. Tabulate the results in Table 3.

Table 3

S.No.	Digital Input (HEX)	Theoretical Analog voltage (volts)	D/A converter output (volts)
1	00	0	0
2	20	0.625	0.75
3	40	1.250	1.25
4	60	1.875	1.87
5	80	2.500	2.5
6	A0	3.125	3.12
7	C0	3.750	3.75
8	E0	4.375	4.37
9	FF	5.000	5.0

Generation of waveforms using D/A converter : The same circuit diagram can be used to generate different waveforms by executing the following programs.

A. Square/Rectangular Wave Generation

Software: The following steps are involved in the generation of square/rectangular wave.

a. Load accumulator with 00H.
b. Output the contents of accumulator to port-1
c. Call delay subroutine
d. Load accumulator with FFH
e. Output the contents of accumulator to port-1
f. Call delay subroutine
g. Repeat from step (b).

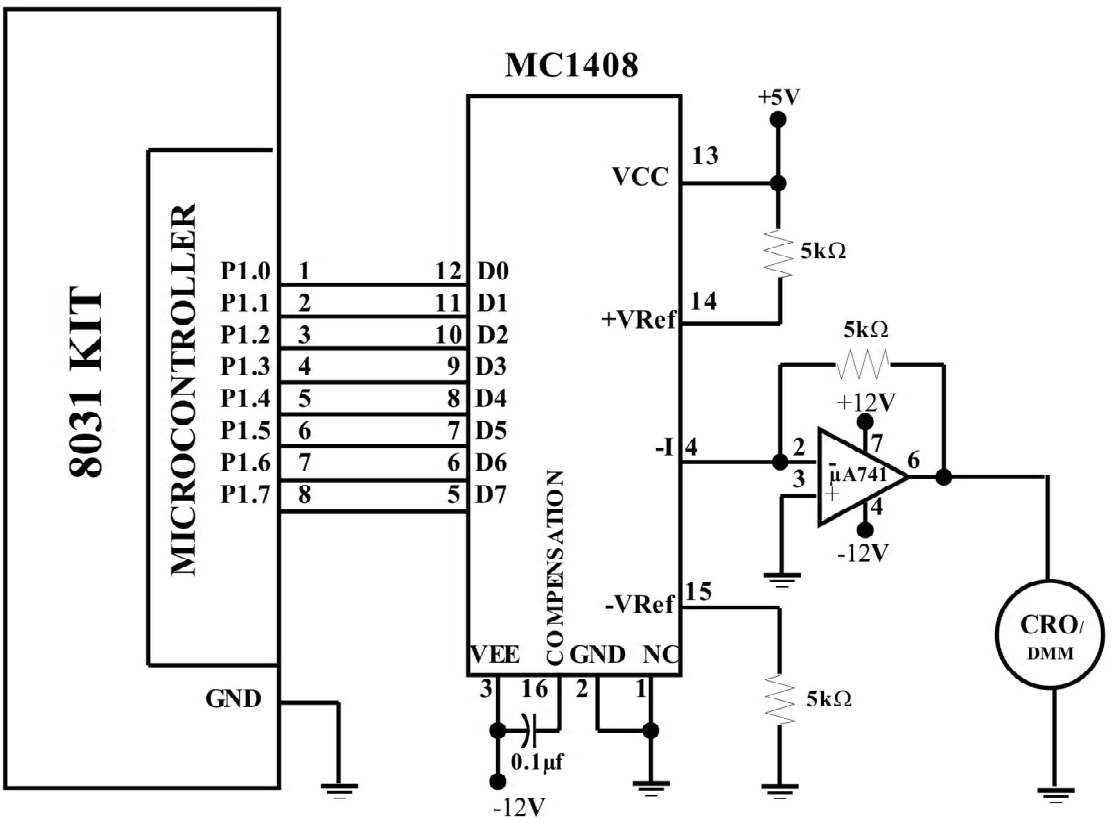


Fig. 17. Interfacing of D/A converter with microcontroller.

Procedure

1. Load the assembly language program -17 (or embedded 'C program) into program memory from the starting address 8000H of the microcontroller kit.
2. Execute the program. Observe the square wave at the output of D/A converter on Cathode Ray Oscilloscope (CRO).
3. Measure the frequency of the square wave. The frequency of the wave form can be varied by varying the delay.
4. Load r0 contents of Delay1 subroutine different from r0 contents of delay2.
5. Execute the program. Observe the rectangular wave on the CRO.

B. Triangular Wave Generation

Software: The following steps are involved in the generation of triangular wave.

a. Load accumulator with 00H
b. Output the accumulator contents to port-1

c. Increment the contents of the accumulator by one
d. Compare the accumulator contents with FFH. If not equal, repeat the above steps from (c), if equal execute step (f)
e. Output the contents of accumulator to port-1
f. Decrement accumulator contents by one
g. Compare the accumulator contents with 00H. If not equal, repeat the program from step (e), If equal repeat from step (c).

Procedure

1. Load the assembly language program-18 (or embedded 'C program) into program memory of microcontroller kit (from the starting address from 8000H).
2. Execute the program. Observe the triangular wave at the output of D/A converter on CRO.
3. Measure the frequency of the triangular wave. The frequency of the waveform can be varied by changing the delay.

C. Ramp/Saw-tooth Wave Generation

Software: The following steps are involved in the generation of saw-tooth wave.
a. Clear accumulator
b. Output contents of accumulator to port-1
c. Call delay subroutine
d. Increment the accumulator contents by one
e. Repeat steps from (b).

Procedure

1. Load the assembly language program-19 (or embedded 'C program) into program memory of microcontroller kit (from the starting address from 8000H).
2. Execute the program. Observe the saw-tooth wave at the output of D/A converter on CRO.
3. Measure the frequency of the saw-tooth wave. The frequency of the waveform can be varied by changing the delay.

Program 16 : Program to convert digital data into analog voltage using D/A converter MC1408

Address	Opcode	Label	Mnemonic	Operand	Comments
8000	74 10	mov		a,#10h	;load acc. with digital data ;to be converted into analog ;voltage
8002	F5 90	mov		p1,a	;send to the DAC
8004	80 FE	sjmp		$	;stay in infinite loop
8006	end				;program ends

Program 17 : Program to generate ramp wave using D/A converter MC1408

Address	Opcode	Label	Mnemonic	Operand	Comments
8000	E4		clr	a	;clear accumulator
8001	F5 90	x1:	mov	p1,a	;send accumulator contents to ;DAC
8003	12 80 0A		call	delay	;call delay subroutine
8006	04		inc	a	;increments contents of ;accumulator by one
02	80 01		ljmp	x1	;jump to location x1 for ;continuous Ramp wave ;generation
					;delay sub-program
800A	78 02	delay:	mov	r0,#02h	
800C	79 FF	l1:	mov	r1,#0ffh	
800E	D9 FE		djnz	r1,$	
8010	D8 FA		djnz	r0,l1	
8012	22	ret			
8013		end			;program ends

Program 18 : Program to generate square/rectangular wave using D/A converter MC1408

Address	Opcode	Label	Mnemonic	Operand	Comments
8000	74 00	x1:	mov	a,#00h	;load accumulator with 00h
8002	F5 90		mov	p1,a	;send to port-1
8004	12 80 0F		lcall	delay1	;call dealy1 sub-program
8007	74 FF		mov	a,#0ffh	;load accumulator with 0ffh
8009	12 80 18		lcall	delay2	;call delay2 sub-program
800C	02 80 00		ljmp	x1	;jump to location x1
					;delay1 sub-program
800F	78 05	delay1:	mov	r0,#05h	
8011	79 FF	l1:	mov	r1,#0ffh	
8013	D9 FE		djnz	r1,$	
8015	D8 FA		djnz	r0,l1	
8017	22		ret		
					;delay2 sub-program
8018	78 05	delay2:	mov	r0,#05h	
801A	79 FF	l2:	mov	r1,#0ffh	
801C	D9 FE		djnz	r1,$	
801E	D8 FA		djnz	r0,l2	
8020	22		ret		
8021			end		;program ends

Program 19 : Program to generate triangular wave using D/A converter MC1408

Address	Opcode	Label	Mnemonic	Operand	Comments
		begin:			
8000	74 00	mov		a,#00h	;load acc. with 00H
8002	F5 90	p_slope:	mov	p1,a	;send to DAC
8004	12 80 17	lcall		delay	;call delay sub-program
8007	04	inc		a	;increment content of acc.
8008	B4 FF F7	cjne		a,#0ffh,p_slope	;decrement acc. by one ;if # 00h then send acc. to ;DAC else go to next step
800B	F5 90	n_slope:	mov	p1,a	;send acc. contents to DAC
800D	12 80 17	lcall		delay	;call delay sub-program
8010	14	dec		a	;decrement acc. by one
8011	B4 00 F7	cjne		a,#00h,n_slope	;if acc. # 00h then send ;acc. to DAC
8014	02 80 00	ljmp		begin	;go to beginning of the ;program
8017	78 02	delay:	mov	r0,#02h	
8019	79 FF	l1:	mov	r1,#0ffh	
801B	D9 FE	djnz		r1,$	
801D	D8 FA	djnz		r0,l1	
801F	22	ret			
8020		end			;program ends

Experiment 16

AIM : To construct an A/D converter using D/A converter by successive approximation technique.

COMPONENTS: D/A converter MC1408, Op-amp μA741, Op-amp LM324, 3-5.1kΩ resistors, bread board and connecting wires.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A, ±12V/100mA

INTRODUCTION: Successive approximation analog to digital (A/D) converter consists of three major elements viz., D/A converter, successive approximation register and comparator. The conversion technique involves comparing the output of D/A converter with the analog input signal which is to be digitized. The digital input to the D/A converter is generated using the successive approximation method. When the D/A converter output matches the analog signal, the digital data applied to D/A converter input can be taken as equivalent digital data. The microcontroller plays the role of successive approximation register.

Hardware: The interfacing circuit for the implementation of analog to digital conversion using the D/A converter is shown in Fig. 18. Port-1 of the microcontroller is connected to the input data lines of D/A converter MC1408. Analog input is given to the non-inverting input of op-amp LM324 which is used as comparator. The D/A converter output is given to the another input of the comparator. The output of the comparator is connected to P3.4 of microcontroller. Logic '1' of the comparator indicates that the D/A converter output is less than the analog input. Similarly logic '0' of the comparator indicates that the D/A converter output is greater than the analog input.

Software

1. Select port-1 of microcontroller as output port to send 8-bit successive approximation value to D/A converter.
2. Send digital data 00H to the D/A converter
3. Set P1.7 pin of microcontroller i.e. MSB of D/A converter
4. Call delay

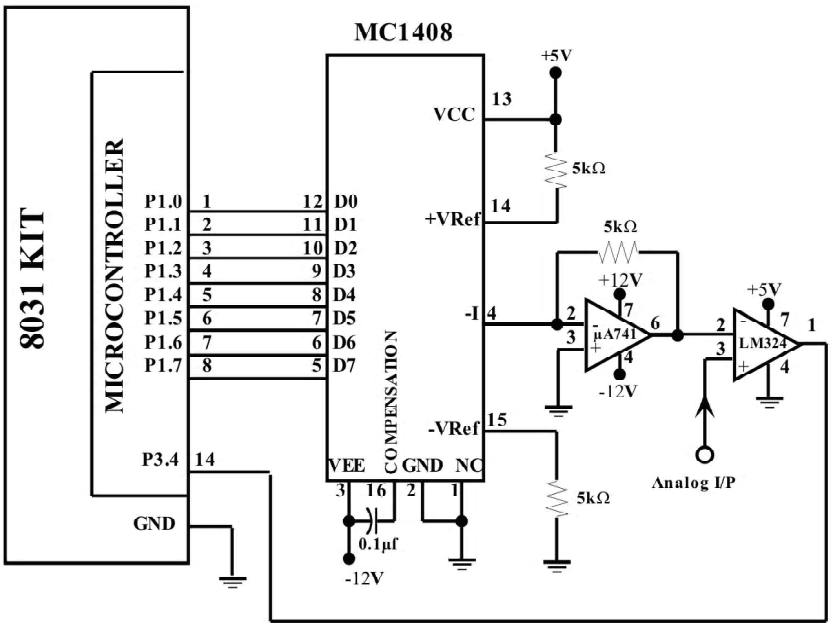


Fig. 18. The interfacing circuit of analog to digital conversion using the D/A converter.

5. Read the output of the comparator through P3.4.
6. If the comparator output is logic 1 then clear P1.7 of microcontroller and set P1.6. If the comparator output is logic 0 then retain P1.7 as logic 1 and set P1.6
7. Call delay
8. Read the output of the comparator through P3.4
9. If comparator output is logic 1 then clear P1.6 of microcontroller and set P1.5. If the comparator output is logic 0 then retain P1.6 as logic 1 and set P1.5
10. Call delay
11. Read the output of the comparator through P3.4.
12. If comparator output is logic 1 then clear P1.5 of microcontroller and set P1.4. If the comparator output is logic 0 then retain P1.5 as logic 1 and set P1.4
13. Call delay
14. Read the output of the comparator through P3.4
15. If comparator output is logic 1 then clear P1.4 of microcontroller and set P1.3. If the comparator output is logic 0 then retain P1.4 as logic 1 and set P1.3
16. Call delay
17. Read the output of the comparator through P3.4.
18. If comparator output is logic 1 then clear P1.3 of microcontroller and set P1.2. If the comparator output is logic 0 then retain P1.3 as logic 1 and set P1.2
19. Call delay
20. Read the output of the comparator through P3.4
21. If comparator output is logic 1 then clear P1.2 of microcontroller and set P1.0. If the comparator output is logic 0 then retain P1.2 as logic 1 and set P1.0

22. Call Delay
23. If comparator output is logic 1 than clear P1.1 of microcontroller and set P1.0 if comperator output is logic 0 then retain P1.1 as logic 1 & set P1.0.
24. Call delay
25. Read the output of the comparator through P3.4
26. If comparator output is logic 1 then clear P1.0 of microcontroller else retain logic 1 on P1.0.
27. Read contents of port-1 of microcontroller to accumulator.
28. Copy the contents of accumulator to r6 register.
29. Display the contents of the r6 register by calling subroutine (0670H).
30. Repeat the above steps to display digital data continuously.

Procedure

1. Wire the circuit as shown in Fig. 18.
2. Load the assembly language program -20 (or embedded 'C program) into the program memory from the starting address 8000H of microcontroller kit.
3. Execute the program. Observe the continuous display of the digital equivalent of analog input voltage on data field of the kit.
4. Vary the analog input voltage from 0-5V in steps of 0.5V and record the corresponding digital data.
5. Tabulate the results in Table 4.

Table 4

S.No	Analog voltage i/p (Volts)	Digtal value (HEX)
1	0	00
2	0.025	01
3	0.04	02
4	1.000	32
5	1.500	4B
6	2.000	64
7	2.255	70
8	3.762	BC
9	4.037	CA
10	4.618	E6
11	4.999	FE

Program 20 : Program for construction of A/D converter using D/A converter

Address	Opcode	Label	Mnemonic	Operand	Comments
8000	E4	l1:	clr	a	;clear accumulator
8001	F5 90	mov		p1,a	;send to DAC
8003	D2 97	setb		p1.7	;set D7 of DAC
8005	12 80 5C	call		delay	;call subroutine for small delay
8008	30 B4 02	jnb		p3.4,x1	;poll output of comparator, if ;comparator output is LOW then

Program 20 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
800B	C2 97	clr		p1.7	;clear D7 of DAC
800D	D2 96	x1: setb		p1.6	;else retain D7 and set D6
800F	12 80 5C	call		delay	;call delay subroutine
8012	30 B4 02	jnb		p3.4,x2	;poll output of comparator, if ;comparator output is LOW then
8015	C2 96	clr		p1.6	;clear D6 of DAC
8017	D2 95	x2: setb		p1.5	;else set D5 of DAC
8019	12 80 5C	call		delay	;call delay subroutine
801C	30 B4 02	jnb		p3.4,x3	;poll output of comparator, if ;comparator output is LOW then
801F	C2 95	clr		p1.5	;clear D5 of DAC
8021	D2 94	x3: setb		p1.4	;else set D4 of DAC
8023	12 80 5C	call		delay	;call delay subroutine
8026	30 B4 02	jnb		p3.4,x4	;poll output of comparator, if ;comparator output is LOW then
8029	C2 94	clr		p1.4	;clear D4 of DAC
802B	D2 93	x4: setb		p1.3	;else set D3 of DAC
802D	12 80 5C	call		delay	;call delay subroutine
8030	30 B4 02	jnb		p3.4,x5	;poll comparator output, if ;comparator output is LOW then
8033	C2 93	clr		p1.3	;clear D3 of DAC
8035	D2 92	x5: setb		p1.2	;else set D2 of DAC
8037	12 80 5C	call		delay	;call delay subroutine
803A	30 B4 02	jnb		p3.4,x6	;poll comparator output, if ;comparator output is LOW then
803D	C2 92	clr		p1.2	;clear D2 of DAC
803F	D2 91	x6: setb		p1.1	;else set D1 of DAC
8041	12 80 5C	call		delay	;call delay subroutine
8044	30 B4 02	jnb		p3.4,x7	;poll comparator output, if ;comparator output is LOW then
8047	C2 91	clr		p1.1	;clear D1 of DAC
8049	D2 90	x7: setb		p1.0	;else set D0 of DAC
804B	12 80 5C	call		delay	;call delay subroutine
804E	30 B4 02	jnb		p3.4,x8	;poll comparator output, if ;comparator output is LOW then
8051	C2 90	clr		p1.0	;clear D0 of DAC
8053	E5 90	x8: mov		a,p1	;read port-1 data
8055	FE	mov		r6,a	;copy digital data to r6
8056	12 06 70	lcall		0670h	;call subroutine to display ;digital data in data field of kit
8059	02 80 00	ljmp		l1	;repeat the above program ;continuously
					;delay program
805C	78 05	delay:	mov	r0,#05	
805E	79 FF	p1 mov		r1,#0ffh	
8060	D8 FE	djnz		r0,$	
8062	D9 FA	djnz		r1,p1	
8064	22	ret			
8065		end			;program ends

Experiment 17

AIM: To interface an 8-bit analog to digital (A/D) converter with microcontroller.

COMPONENTS: Analog to digital converter ADC0809, Op-amp LM308, NE555 timer, precision voltage regulator LM329, 2-1kΩ resistors, 2-10kΩ pots, 10μF, 0.1μF, 0.01μF, capacitors, bread board, and connecting wires.

EQUIPMENT: Microcontroller kit, power supply +5V/3A, ±12V/500mA

INTRODUCTION: The analog to digital conversion is a quantizing process in which an analog signal is represented by equivalent binary states. This is opposite to digital to analog conversion process. A/D converters can be classified into two general groups based on the conversion technique. One technique involves comparing a given signal with the internally generated equivalent signal. The second technique involves conversion of an analog signal into time or frequency and comparing these two parameters against known values. The choice between the two techniques is based on accuracy versus speed.

The ADC0809 is monolithic CMOS device with an 8-bit A/D converter, 8 channel multiplexer and microprocessor/microcontroller control logic. It uses successive approximation as the conversion technique. The design of the ADC0809 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. It offers high speed, high accuracy, minimal temperature dependence, excellent long term accuracy and repeatability. It consumes less power. These features make the device ideally suited for process and machine control, consumer, and automobile applications.

The address lines ADDA, ADDB, & ADDC are used to select the analog signal applied to one of the eight input channels. The following table gives the address on the address lines and channel section.

Table 5

ADDC	ADDB	ADDA	Channel selected
0	0	0	channel-0
0	0	1	channel-1
0	1	0	channel-2
0	1	1	channel-3
1	0	0	channel-4
1	0	1	channel-5
1	1	0	channel-6
1	1	1	channel-7

The address latch enable (ALE), start of conversion (SOC) and end of conversion (EOC) are control lines. In order to initiate the process of analog to digital conversion, the following sequence of control signals must be sent to the A/D converter.

a. 3-bit channel address of the desired channel
b. After at least 50 μsec. the ALE input must be sent HIGH
c. After at least 2.5 μsec. the SOC input must be sent HIGH and then LOW
d. Then ALE input should be brought LOW
e. When the EOC signal from A/D converter is found HIGH. The 8-bit digital data will be available on data lines of A/D converter.

Hardware: Figure 19 shows the circuit diagram of interfacing of A/D converter ADC0809 with microcontroller. In the present experiment the analog voltage is applied

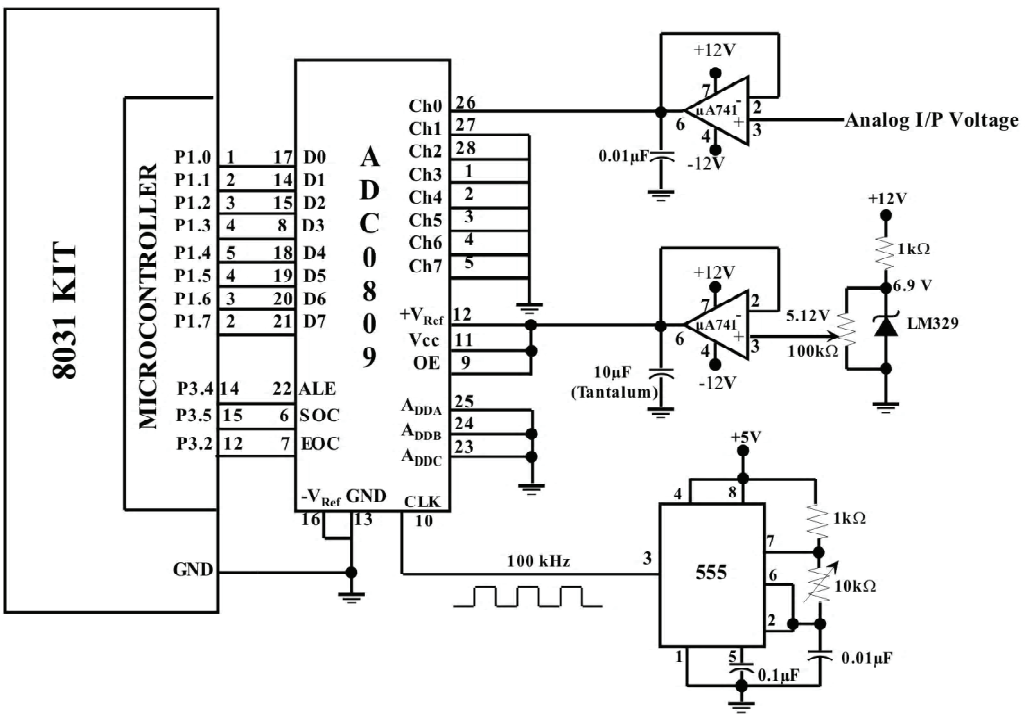


Fig. 19. The circuit diagram of interfacing of A/D converter ADC0809 with microcontroller.

to channel-0, the channel selecting lines ADDA, ADDB & ADDC are connected to the ground to select channel-0. The remaining channels lines are connected to the ground. The control lines ALE, SOC & EOC of A/D converter are connected to P3.4, P3.5 and P3.2 respectively. The digital output lines D0-D7 of the converter are connected to port-1 of the microcontroller. A clock frequency of 100 kHz is generated using NE555 timer and is fed to the clock pin of A/D converter. The output enable pin of A/D converter is connected to +5V which enables the internal data bus latch. The analog voltage is applied to the channel-0 of the A/D converter. A stable 5.12V is generated using precision voltage regulator LM329 and buffer, this voltage is applied to Vref and Vcc of the A/D converter. The 5.12V reference voltage makes the resolution of A/D converter as 20mV.

Software: The following steps are involved in the process of conversion of analog voltage into the corresponding digital data.

a. Send HIGH on ALE and wait for few microseconds
b. Send HIGH to SOC and wait for few microseconds
c. Send LOW on SOC
d. Send LOW on ALE
e. Read the EOC through P3.2 till it goes to HIGH
f. Read the digital data from A/D converter through port-1
g. Copy the contents of accumulator to r6 register
h. Display the contents of the r6 register on data field of the kit by calling the subroutine
i. Repeat the above steps to display digital data continuously.

Procedure

1. Wire the circuit as shown in Fig. 19.
2. Load the program-21 into the program memory from the starting address 8000H of microcontroller kit.
3. Execute the program. Observe the continuous display of the digital equivalent of analog input voltage on data field of the kit.
4. Vary the analog input voltage from 0-5V in steps of 0.5V and record the corresponding digital data.
5. Tabulate the results in Table 6.

Table 6

S.No.	Analog voltage (Volts)	Digtal value (HEX)
1	0.00	00
2	0.48	18
3	1.75	55
4	2.02	66
5	2.56	7F

Table 6: Contd

S.No.	Analog voltage (Volts)	Digtal value (HEX)
6	3.13	9C
7	3.41	AA
8	4.75	ED
9	5.00	FA
10	5.12	FF

Program 21: Program for analog voltage into digital conversion through ADC0809

Address	Opcode	Label	Mnemonic	Operand	Comments
8000	D2 B4	setb		p3.4	;send ALE HIGH
8002	00	nop			;small delay
8003	00	nop			
8004	00	nop			
8005	00	nop			
8006	00	nop			
8007	00nop				
8008	00	nop			
8009	00	nop			
800A	D2 B5	setb		p3.5	;send SOC HIGH
800C	00	nop			
800D	00nop				
800E	00nop				
800F	00	nop			
8010	C2 B5	clr		p3.5	;send SOC LOW
8012	C2 B4	clr		p3.4	;send ALE LOW
8014	30 B2 FD	jnb		p3.2, $	;poll EOC until it ;becomes HIGH
8017	E5 90	mov		a,p1	;read digital data through port-1
8019	FE	mov		r6,a	;store digital data in r6
801A	12 06 C0	lcall		06c0h	;call subroutine to display ;the contents of r6 on the ;data field of microcontroller kit ;(this address may vary from kit ;to kit)
801D		end			;program ends

Experiment 18

AIM: To interface liquid crystal display (LCD) module with microcontroller and display labels on it.

COMPONENTS: 16 × 2 liquid crystal display module, 20kΩ pot, bread board and connecting wires.

EQUIPMENT: Microcontroller kit, power supply + 5V/3A.

INTRODUCTION: The Oriole's Display Module, ODM (2 line × 16 characters) is used in the present experiment. It is a dot matrix liquid crystal display (LCD) which displays English alphanumeric, Kana (Japanese) characters and symbols. The interfacing of the liquid crystal display module with the microcontroller helps in making the instrument portable, readable with a less power consumption.

Hardware: The circuit diagram shown in Fig. 20 presents the interfacing of the display module with the microcontroller. In the present experiment a 2 line × 16 character LCD module is used. The data bus D_0-D_7 of display module is connected to P1.0 – P1.7 of the microcontroller and the control pins RS and E are connected to the P3.2 and P3.3 pins of microcontroller respectively. The command word is sent to the command register by making RS LOW through P3.2 and the data can be sent to the display RAM by making RS HIGH through P3.2. A pulse of about 40 microseconds duration is sent to the pin E of the microcontroller to fetch/send the data/command word to the display, which is done through the pin P3.3 of the microcontroller. The display read/write pin R/W is connected to the ground so that, the display will be in write mode. The second pin of the display is connected to the variable terminal of the potentiometer to vary the contrast of the display.

Software

1. Select port-1 of microcontroller to read/write digital data/control word.
2. Select port-3 bits (P3.2 - P3.3) of microcontroller for sending RS and enable E pulse.

3. Send the function set control word (38H) to LCD for 8-bit data length, 2-line display and 5×7 character font.
4. Call subroutine to generate RS = 0 and 40 μsec HIGH pulse through P3.2 and P3.3 respectively. This will make the control word to enter into the display controller.
5. Send display ON/OFF control word (0EH) to LCD for display and cursor ON.
6. Call subroutine to generate RS = 0 and 40 μsec HIGH pulse.
7. Send entry mode set control word (06H) to LCD for to shift cursor right after entry of a character.
8. Call subroutine to generate RS = 0 and 40 μsec HIGH pulse.
9. Send clear display word (01H) to LCD for clearing the display.
10. Call subroutine to generate RS = 0 and 40 μsec HIGH pulse.
11. Send return home control word (02H) to LCD for bringing the cursor to left corner of the display.
12. Call subroutine to generate RS = 0 and 40 μsec HIGH pulse.
13. Load DPTR with the starting address of the codes of 'Instrumentation'.
14. Load register r0 with count 0CH for displaying 12 characters on LCD module.
15. Read character code and send to the display.
16. Call subroutine to generate RS = 1 and 40 μsec HIGH pulse through P3.2 and P3.3 respectively. This will make the data to enter into the display controller to display same on the LCD module.
17. Increment DPTR by one to send next character.
18. Decrement r0 contents by one. If r0?00H then repeat above steps from (15).
19. If r0=00H then enter into continuous loop.

Procedure

1. Wire the circuit as shown in Fig. 20.
2. Load the program-22 (assembly or embedded 'C' program) into the program memory from the starting address 8000H and store ASCII codes (49 4E 53 54 52 55 4D 45 4E 54 41 54 49 4F 4E) to display "INSTRUMENTATION "from the starting address 8100H of microcontroller kit.
3. Execute the program. Observe the display of label "INSTRUMENTATION" on LCD module.

   The 'C' programs are written in Raisonance Integrated Development Environment (RIDE).

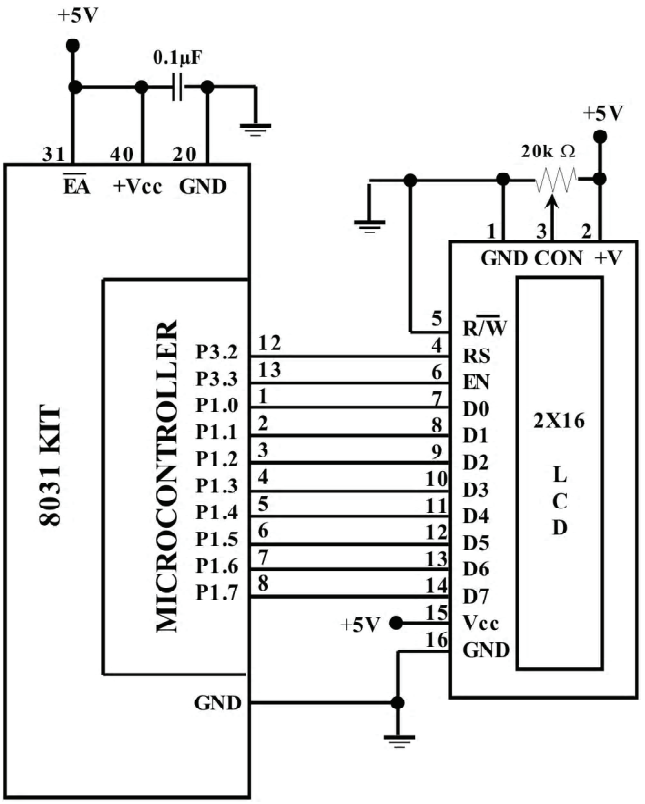


Fig. 20. Interfacing of the LCD module with the microcontroller

Program 22 : Program for displaying label "Instrumentation" on LCD module

Address	Opcode	Label	Mnemonic	Operand	Comments
		org	8100h		
8100	49 4E 53 54 52	label:		db 'I','N','S','T','R','U','M','E','N','T','A','T','I','O','N'	
8105	55 4D 45 4E 54				
810A	41 54 49 4F 4E				
8000		org	8000h		
					;Initialization of LCD display
8000	75 81 40	mov		sp,#40h	;Initialize stack pointer ;with starting address 40h
8003	74 38	mov		a,#38h	;function set control word
8005	12 80 2A	lcall		enable	;send to LCD module and ;generate RS=0, and enable ;pulse
8008	74 0E	mov		a,#0eh	;display ON/OFF control word
800A	12 80 2A	lcall		enable	;send to the LCD module

Program 22 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
800D	74 06	mov		a,#06h	;entry mode set control word
800F	12 80 2A	lcall		enable	;send to the LCD module
8012	74 01	mov		a,#01h	;clear display control word
8014	12 80 2A	lcall		enable	;send to the LCD module
8017	74 02	mov		a,#02h	;return home control word
8019	12 80 2A	lcall		enable	;send to the LCD module
801C	90 81 00	l1:	mov	dptr,#label	;load DPTR with starting ;address of the codes
801F	78 0F	mov		r0,#0fh	;load register r0 with 12
8021	E0	x1:	movx	a,@dptr	;read character code
8022	12 80 3F	lcall		data_disp	;call subroutine to send ;RS=1 and HIGH pulse
8025	A3	inc		dptr	;decrement DPTR to point ;next character
8026	D8 F9	djnz		r0,x1	;decrement r0,if r0!=00H ;then go to x1 location
8028	80 FE	sjmp		$	;stay in continuous loop enable:
802A	F5 90	mov		p1,a	;send control word to display
802C	C2 B2	clr		p3.2	;RS=0
802E	D2 B3	setb		p3.3	;enable=1
8030	75 F0 FF	mov		b,#0ffh	;small delay
8033	D5 F0 FD	djnz		b,$	
8036	C2 B3	clr		p3.3	;enable=0
8038	75 F0 FF	mov		b,#0ffh	;small delay
803B	D5 F0 FD	djnz		b,$	
803E	22	ret			;return to main program data_disp:
803F	F5 90	mov		p1,a	;send character code to ;display
8041	D2 B2	setb		p3.2	;RS=1
8043	D2 B3	setb		p3.3	;enable=1
8045	75 F0 FF	mov		b,#0ffh	;small delay
8048	D5 F0 FD	djnz		b,$	
804B	C2 B3	clr		p3.3	;enable=0
804D	75 F0 FF	mov		b,#0ffh	;small delay
8050	D5 F0 FD	djnz		b,$	
8053	22	ret			;return to main program
8054		end			;program ends

```
/* Embedded 'C' program for LCD module */
#include <stdio.h>
#include <reg51.h>

sbit rs = P3^2;
sbit  e = P3^3;
```

```
void main (void)
{
void lcd_comand(void); // Declaration of LCD Command
                       // Function
Void lcd_data(void);   // Declaration of LCD Data Function
void lcd_int(void);    // Declaration of LCD Initialization
                       // Function
int i;
                       // String to be displayed on LCD
int x[]= {'I','N','S','T','R','U','M', 'E','N',
'T','A','T','I','O','N'};
lcd_int();             // Initialize LCD
x1:
for (i = 0;i<15;i++)
  {
  P1 = x[i];           // Get the first char in the string
  lcd_data();          // Call Data function
}
  P1 = 0x02;
  lcd_comand();

  x1: goto x1;
}
void lcd_int(void)     // Definition of lcd_int()
  {
  P1 = 0x38;           // Function set command word
  lcd_comand();

  P1 = 0x0e;           // Display on/off command word
  lcd_comand();

  P1 = 0x06;           // Entry mode set command word
  lcd_comand();

  P1 = 0x01;           // Clear display
  lcd_comand();

  P1 = 0x02;           // Cursor home
  lcd_comand();
}
void lcd_comand(void)
  {
  int i;
```

```
 rs = 0;              // Select command register
 e = 1;               // Enable pulse high & delay
 for (i = 1;i<1000;i++);
 e = 0;               // Enable pulse low & delay
 for (i = 1;i<1000;i++);
}

 Void lcd_data(void)
 {

 int i;
 rs = 1;              // Select data register
 e = 1;               // Enable pulse high & delay
 for (i = 1;i<1000;i++);
 e = 0;               // Enable pulse low & delay
 for (i = 1;i<1000;i++);
 }
```

Experiment 19

AIM: 1. To interface an 8-bit A/D converter ADC0809 with microcontroller and
2. To measure the analog voltage and display the same on LCD module.

COMPONENTS: Microcontroller AT89C51, analog to digital converter ADC0809, Op-amp LM308, 555 timer, precision voltage regulator LM329, 2-1kΩ resistors, 2-10kΩ potentiometers, 100Ω resistor, 20 kΩ potentiometer, 10μF, 0.1μF, 0.01μF capacitors, switch, LCD module, bread board and connecting wires.

EQUIPMENT: Power supply +5V/3A, ±12V/100mA

HARDWARE: Figure 21 shows the circuit diagram of interfacing of A/D converter ADC0809 and LCD module with microcontroller AT89C51. In the present experiment the analog voltage to be digitized is applied to the channel-0 of the ADC, hence the address lines ADDA, ADDB, & ADDC are connected to the ground to select channel-0. The remaining channel lines are connected to the ground. The control lines ALE, SOC, & EOC of A/D converter are connected to P2.3, P2.4 and P2.5 of microcontroller respectively. The digital output lines D0-D7 of the converter are connected to port-1 of the microcontroller. A clock frequency of 100 kHz is generated using NE555 timer and is fed to the clock pin of A/D converter. The output enable pin of A/D converter is connected to +5V which enables the internal data bus latch. The analog voltage is applied to the channel-0 of the A/D converter. A stable 5.12V is generated using buffer and precision voltage regulator LM329, this voltage is applied to Vref and Vcc, and OE pins of the A/D converter. The 5.12V reference voltage makes the resolution of A/D converter as 20mV.

The circuit diagram also presents the interfacing of the LCD display module with the microcontroller. The data bus D0-D7 of display module is connected to P1.0-P1.7 of microcontroller and the control pins RS and E are connected to the P3.2 and P3.3 pins of 89C51 respectively.

SOFTWARE: The following steps are involved in the process of conversion of analog voltage into the corresponding digital data.

a. Initialize stack pointer
b. Initialize LCD module
c. Call subroutine to display 'Voltage=' on LCD module
d. Call A/D conversion subroutine
e. Call subroutine to multiply ADC data with C4H(5/255)
f. Call subroutine to convert HEX to BCD
g. Call subroutine to convert BCD to ASCII code and display voltage on LCD module
h. Call delay subroutine
i. Send control word to the LCD module to bring the cursor to left corner of the module
j. Repeat the above steps to display digital data continuously.

Procedure

1. Wire the circuit as shown in Fig. 21.
2. Load the assembly language program-23 or 'C' program into the program memory from the starting address 0000H.
3. Execute the program by powering the circuit. Observe the display of the analog input voltage on LCD module
4. Vary the analog input voltage from 0-5V in steps of 0.5V and record the corresponding digital data.
5. Tabulate the results in Table 7.

Table 7

S.No.	Analog voltage DVM (Volts)	Voltage Displayed on LCD module
1	0.00	0.00
2	1.11	1.10
3	1.99	1.98
4	2.34	2.34
5	2.61	2.60
6	3.07	3.06
7	3.75	3.74
8	4.26	4.26
9	4.89	4.88
10	5.12	5.12

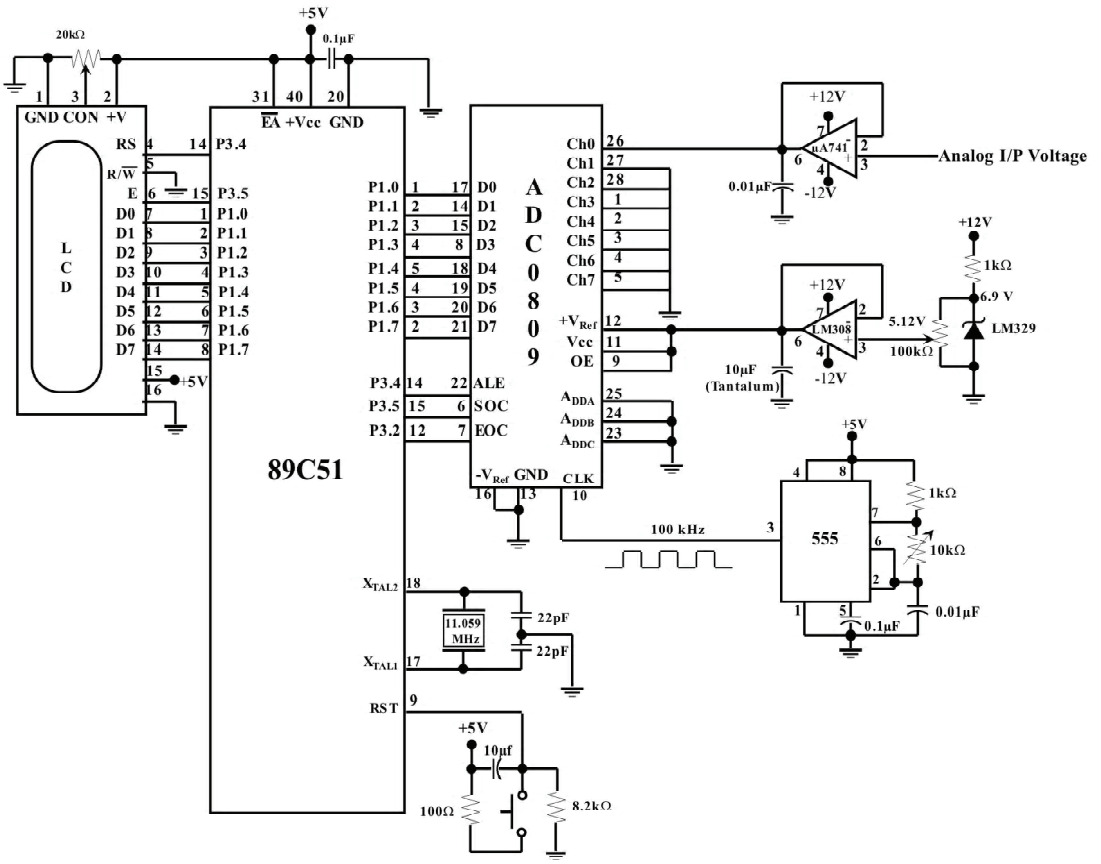


Fig. 21. Interfacing of A/D converter ADC0809 and LCD module with microcontroller AT89C51.

Program 23: Program for measuring analog voltage using ADC0809 and displaying on LCD module

Address	Opcode	Label	Mnemonic	Operand	Comments
		org	0000h		;starting address of the program ;Initialization of LCD display
0000	75 81 40	mov		sp,#40h	;Initialize stack pointer ;with starting address 40h
0003	74 38	mov		a,#38h	;function set control word
0005	12 00 50	lcall		enable	;send to LCD module
0008	74 0E	mov		a,#0eh	;display ON/OFF control word
000A	12 00 50	lcall		enable	;send to LCD module
000D	74 06	mov		a,#06h	;entry mode set control word
000F	12 00 50	lcall		enable	;send to LCD module
0012	74 01	mov		a,#01h	;clear display control word

Program 23 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0014	12 00 50	lcall		enable	;send to LCD module
0017	74 02	mov		a,#02h	;return home control word
0019	12 00 50	lcall		enable	;send to LCD module
					rep:
001C	12 00 7A	lcall		v_display	;call subroutine to display
					;'Voltage=' on LCD module
001F	12 00 39	lcall		adc	;call A/D subroutine
0022	75 F0 C4	mov		b,#0c4h	;multiplication factor
0025	A4	mul		ab	;(digital data)*(5/255)
0026	F8	mov		r0,a	;copy low byte of result in r0
0027	A9 F0	mov		r1,b	;copy high byte of result in r1
0029	7A 00	mov		r2,#00h	;clear r2 register
002B	12 00 C8	lcall		hex_bcd	;call subroutine to convert
					;HEX number into BCD number
002E	12 00 A3	lcall		display	;call subroutine to display
					;voltage in terms of V
0031	74 02	mov		a,#02h	;bring cursor to left
0033	12 00 50	lcall		enable	;corner of the display
0036	02 00 1C	ljmp		rep	;repeat the program
0039		adc:			
0039	D2 B4	setb		p2.3	;send ALE HIGH
003B	00	nop			;small delay
003C	00	nop			
003D	00	nop			
003E	00	nop			
003F	00	nop			
0040	00	nop			
0041	D2 B4	setb		p2.4	;send SOC HIGH
0043	00	nop			
0044	00	nop			
0045	00	nop			
0046	C2 B4	clr		p2.4	;send SOC LOW
0048	C2 B2	clr		p2.3	;send ALE LOW
004A	30 B3 FD	jnb		p2.5,$	;poll EOC until it
					;becomes HIGH
004D	E5 80	mov		a,p0	;read digital data through port-1
004F	22			ret	;return to main program enable:
0050	F5 90	mov		p1,a	;send control word to display
0052	C2 B2	clr		p3.2	;RS=0
0054	D2 B3	setb		p3.3	;enable=1
0056	75 F0 FF	mov		b,#0ffh	;small delay
0059	D5 F0 FD	djnz		b,$	
005C	C2 B3	clr		p3.3	;enable=0
005E	75 F0 FF	mov		b,#0ffh	;small delay
0061	D5 F0 FD	djnz		b,$	
0064	22	ret			;return to main program
					data_disp:

Program 23 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0065	F5 90	mov		p1,a	;send character code to display
0067	D2 B2	setb		p3.2	;RS=1
0069	D2 B3	setb		p3.3	;enable=1
006B	75 F0 FF	mov		b,#0ffh	;small delay
006E	D5 F0 FD	djnz		b,$	
0071	C2 B3	clr		p3.3	;enable=0
0073	75 F0 FF	mov		b,#0ffh	;small delay
0076	D5 F0 FD	djnz		b,$	
0079	22	ret			;return to main program
					v_display:
007A	74 56	mov		a,#56h	;code for 'V'
007C	12 00 65	lcall		data_disp	;display it on LCD module
007F	74 6F	mov		a,#6fh	;code for 'o'
0081	12 00 65	lcall		data_disp	;display it on LCD module
0084	74 6C	mov		a,#6ch	;code for 'l'
0086	12 00 65	lcall		data_disp	;display it on LCD module
0089	74 74	mov		a,#74h	;code for 't'
008B	12 00 65	lcall		data_disp	;display it on LCD module
008E	74 61	mov		a,#61h	;code for 'a'
0090	12 00 65	lcall		data_disp	;display it on LCD module
0093	74 67	mov		a,#67h	;code for 'g'
0095	12 00 65	lcall		data_disp	;display it on LCD module
0098	74 65	mov		a,#65h	;code for 'e'
009A	12 00 65	lcall		data_disp	;display it on LCD module
009D	74 3D	mov		a,#3dh	;code for '='
009F	12 00 65	lcall		data_disp	;display it on LCD module
00A2	22	ret			;return to main program
			display:		
00A3	EB	mov		a,r3	;read most significant digit
00A4	44 30	orl		a,#30h	;convert it into ASCII
00A6	12 00 65	lcall		data_disp	;display it on LCD
00A9	74 2E	mov		a,#2eh	;load acc. with code for '.'
00AB	12 00 65	lcall		data_disp	;display it on LCD
00AE	EC	mov		a,r4	;read next digit
00AF	44 30	orl		a,#30h	;convert it into ASCII
00B1	12 00 65	lcall		data_disp	;display it on LCD
00B4	ED	mov		a,r5	;read next digit
00B5	44 30	orl		a,#30h	;convert it into ASCII
00B7	12 00 65	lcall		data_disp	;display it on LCD
00BA	22	ret			;return to main program
			delay:		
00BB	7D 04	mov		r5,#04h	
00BD	7F FF	l8:	mov	r7,#0ffh	
00BF	7E FF	l7:	mov	r6,#0ffh	
00C1	DE FE	djnz		r6,$	
00C3	DF FA	djnz		r7,l7	
00C5	DD F6	djnz		r5,l8	
00C7	22	ret			

Program 23 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
					;input 24 bit HEX number is in r2_hb,r1,r0_lb
					;output BCD number is in r0_hb,r1,r2,r3,r4,r5,r6,r7_lb
		hex_bcd:			
00C8	7F 19		mov	r7,#19h	
00CA	12 01 29		lcall	conv1	
00CD	EB		mov	a,r3	
00CE	FC		mov	r4,a	
00CF	7F 19		mov	r7,#19h	
00D1	12 01 29		lcall	conv1	
00D4	EB		mov	a,r3	
00D5	FD		mov	r5,a	
00D6	7F 19		mov	r7,#19h	
00D8	12 01 29		lcall	conv1	
00DB	EB		mov	a,r3	
00DC	FE		mov	r6,a	
00DD	7F 19		mov	r7,#19h	
00DF	12 01 29		lcall	conv1	
00E2	EB		mov	a,r3	
00E3	FF		mov	r7,a	
00E4	EC		mov	a,r4	
00E5	54 0F		anl	a,#0fh	
00E7	F8		mov	r0,a	
00E8	EC		mov	a,r4	
00E9	54 F0		anl	a,#0f0h	
00EB	C4		swap	a	
00EC	F9		mov	r1,a	
00ED	ED		mov	a,r5	
00EE	54 0F		anl	a,#0fh	
00F0	FA		mov	r2,a	
00F1	ED		mov	a,r5	
00F2	54 F0		anl	a,#0f0h	
00F4	C4		swap	a	
00F5	FB		mov	r3,a	
00F6	EE		mov	a,r6	
00F7	54 0F		anl	a,#0fh	
00F9	FC		mov	r4,a	
00FA	EE		mov	a,r6	
00FB	54 F0		anl	a,#0f0h	
00FD	C4		swap	a	
00FE	FD		mov	r5,a	
00FF	EF		mov	a,r7	
0100	54 0F		anl	a,#0fh	
0102	FE		mov	r6,a	
0103	EF		mov	a,r7	
0104	54 F0		anl	a,#0f0h	
0106	C4		swap	a	
0107	FF		mov	r7,a	

Program 23 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0108	EF	mov		a,r7	
0109	F5 08	mov		08h,a	
010B	EE	mov		a,r6	
010C	F5 09	mov		09h,a	
010E	ED	mov		a,r5	
010F	F5 0A	mov		0ah,a	
0111	EC	mov		a,r4	
0112	F5 0B	mov		0bh,a	
0114	E8	mov		a,r0	
0115	FF	mov		r7,a	
0116	E9	mov		a,r1	
0117	FE	mov		r6,a	
0118	EA	mov		a,r2	
0119	FD	mov		r5,a	
011A	EB	mov		a,r3	
011B	FC	mov		r4,a	
011C	E5 08	mov		a,08h	
011E	F8	mov		r0,a	
011F	E5 09	mov		a,09h	
0121	F9	mov		r1,a	
0122	E5 0A	mov		a,0ah	
0124	FA	mov		r2,a	
0125	E5 0B	mov		a,0bh	
0127	FB	mov		r3,a	
0128	22	ret			
0129	C3	conv1:	clr	c	
012A	74 00	mov		a,#00h	
012C	FB	mov		r3,a	
012D	C3	conv2:	clr	c	
012E	74 00	mov		a,#00h	
0130	DF 01	djnz		r7, cont	
0132	22	ret			
			cont:		
0133	E8	mov		a,r0	
0134	33	rlc		a	
0135	F8	mov		r0,a	
0136	E9	mov		a,r1	
0137	33	rlc		a	
0138	F9	mov		r1,a	
0139	EA	mov		a,r2	
013A	33	rlc		a	
013B	FA	mov		r2,a	
013C	EB	mov		a,r3	
013D	F5 F0	mov		b,a	
013F	35 F0	addc		a,b	
0141	D4	da		a	
0142	FB	mov		r3,a	
0143	50 E8	jnc		conv2	
0145	E8	mov		a,r0	

Program 23 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0146	34 00	addc		a,#00h	
0148	F8	mov		r0,a	
0149	E9	mov		a,r1	
014A	34 00	addc		a,#00h	
014C	F9	mov		r1,a	
014D	EA	mov		a,r2	
014E	34 00	addc		a,#00h	
0150	FA	mov		r2,a	
0151	02 01 2D	ljmp		conv2	
		end			

```
//program for voltage measurement using ADC0809 A/D converter
#include <stdio.h>
#include <reg51xa.h>

//Assignment of labels for Port-2 pins
   sbit   ada  =  P2^0;
   sbit   adb  =  P2^1;
   sbit   adc  =  P2^2;
   sbit   ale  =  P2^3;
   sbit   soc  =  P2^4;
   sbit   eoc  =  P2^5;

   sbit   rs  =  P3^4;
   sbit    e  =  P3^5;

       int  d,d1,d2,d3;
//main program begins here
   void main  (void)
   {
   float adc_0809(void);
   void lcd_comand(void);
   void lcd_data(void);
   void lcd_int(void);
   void display(void);
   int  i;

   int v[] = {'V','O','L','T','A','G','E','='};
   lcd_int();

   rep1:
```

```
   P1 = 0x02;
   lcd_comand();

   for (i = 0;i<8;i++)
   {
    P1 = v[i];
    lcd_data();
    }
    adc_0809();

    for (i=1;i<9000;i++);
    display();
    goto rep1;
   }
//A/D converter subroutine
   float adc_0809(void)
   {
   int i,x1,x2;
   float v;

   ada = 0;
   adb = 0;
   adc = 0;

   ale = 1;
   for(i=1;i<5;i++);
   soc = 1;
   for(i=1;i<5;i++);
   soc = 0;
   ale = 0;

   do{
   x1 = eoc;
   }while(x1!=1);

   x2 = P0;
   v = ((x2)*5.0)/255;

   v = v*100.0;
   d = v;
   d1 = d/100;

   d2 = d % 100;
```

```
   d2 = d2/10;

   d3 = d % 10;
   }
//display subroutine
   Void display(void)
   {
   d1 = d1+0x30;
   P1 = d1;
   lcd_data();

   P1 = 0x2e;
   lcd_data();

   d2 = d2+0x30;
   P1 = d2;
   lcd_data();

   d3 = d3+0x30;
   P1 = d3;
   lcd_data();
     }

 //LCD initialization subroutine
   void lcd_int(void)
   {
   P1 = 0x38;
   lcd_comand();

   P1 = 0x0e;
   lcd_comand();

   P1 = 0x06;
   lcd_comand();

   P1 = 0x01;
   lcd_comand();

   P1 = 0x02;
   lcd_comand();
   }
```

```
 //RS =0 and E pulse generation subroutine for command word
   void lcd_comand(void)
   {

   int i;
   rs = 0;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
//RS=1 and E pulse generation subroutine for data
   void lcd_data(void)
   {
   int i;
   rs = 1;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
```

Experiment 20

AIM: 1. Interfacing of 12-bit analog to digital converter AD1674 with microcontroller.
2. To measure the analog voltage and display it on LCD module.

COMPONENTS: Microcontroller 89C51, 12-bit A/D converter AD1674, LM308, 12.0 MHz crystal, switch, 2-100Ω, 20kΩ multi-turn pots, 8.2kΩ, 100Ω, 2-1.0μF metalized polyester capacitors, 0.1μF decoupling capacitor, LCD module, bread board and connecting wires.

EQUIPMENT: Power supply + 5V/3A.

INTRODUCTION: The AD1674 is a high speed successive approximation technique A/D converter and it has the following features.

* 12-bit resolution
* 10 μsec conversion time
* Conversion speed up to 100k samples per second
* Does not require external zero and full scale adjustment
* On chip sample and hold facility
* On chip clock facility
* Ranges : bipolar ±5V or ±10V and unipolar 0-10V or 0-20V
* Single power supply

Hardware: Interfacing of A/D converter AD1674 with the microcontroller is presented in Fig. 22. The A/D converter is interfaced through port-0, port-2 and port-3 of microcontroller. The pins R/C and STATUS of analog-to-digital converter are connected to the P3.4 and P3.5 of microcontroller respectively to start conversion and to check whether conversion is completed or not. The circuit diagram also shows the interfacing of the display module with the microcontroller. The data bus D0-D7 of display module is connected to P1.0-P1.7 of microcontroller and the control pins RS and E are connected to the P3.2 and P3.3 pins of 89C51 respectively. Interfacing of LCD module with microcontroller is presented in Experiment-8.

When an analog voltage is applied to the A/D converter input, the process of converting analog voltage into digital signal is started by making R/C pin first HIGH

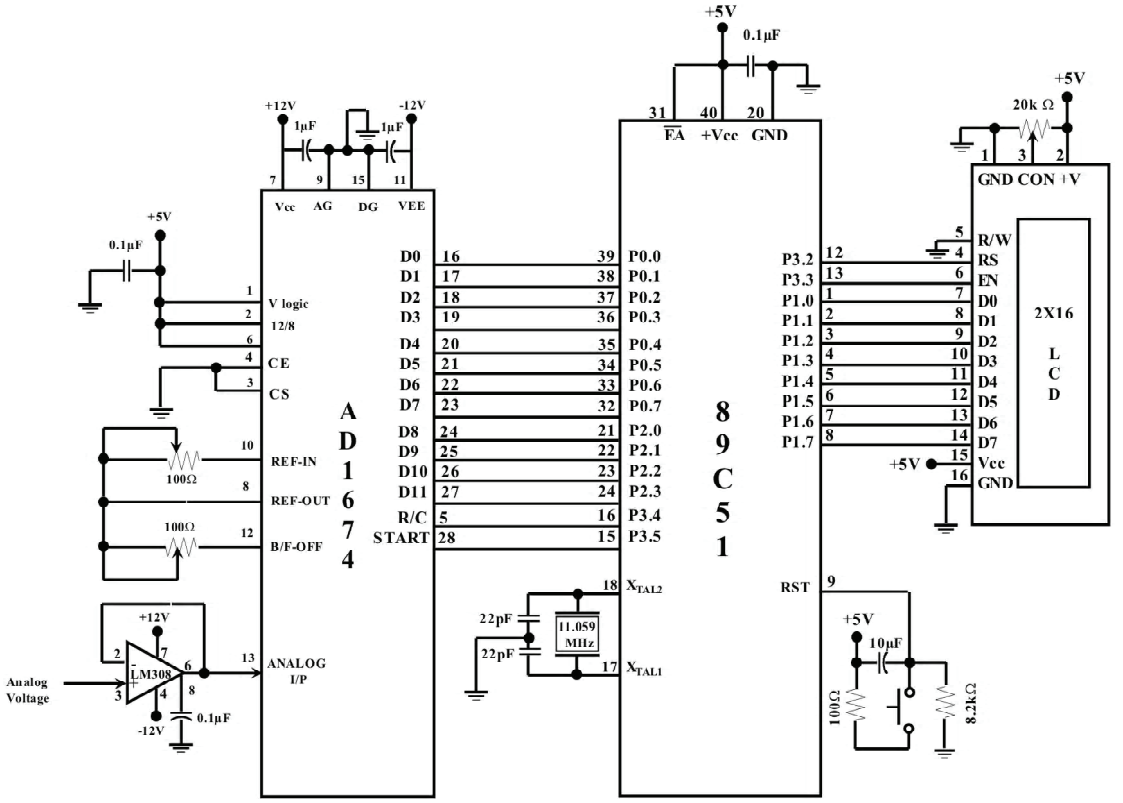


Fig. 22. Interfacing of A/D converter AD1674 with the microcontroller.

and then LOW, this is done by P3.4. To check whether the conversion is completed or not, the STATUS pin is continuously monitored through P3.5. When the STATUS pin goes LOW then conversion is completed and the digital data are available on 12 lines (D0-D11) of A/D converter which is read by microcontroller. The lower byte of digital data is read through port-0 and high nibble of digital data is read through port-2 (P2.0-P2.3). After reading digital data, the microcontroller substitutes this digital data in the equation (3), which gives the input analog voltage directly in volts and the same is displayed on LCD module.

$$V = \frac{\text{Vref* (Digital data)}}{2^{12}} \quad \text{volts} \qquad \text{..... (1)}$$

$$V = (10/4096)*\text{digital data} \quad \text{volts} \qquad \text{..... (2)}$$

$$V = 0.0024414*\text{digital data} \quad \text{volts} \qquad \text{..... (3)}$$

Software

1. Initialize the stack pointer
2. Initialize the LCD module
3. Call subroutine to display 'Voltage=' on LCD module
4. Call subroutine to convert analog voltage into digital data
5. Call subroutine to subtract 7FFH from digital data for bi-polar operation
6. Call subroutine to multiply digital data with 24414 (5E5FH)
7. Call subroutine to convert HEX to BCD number
8. Call subroutine to display voltage on LCD module
9. Call delay subroutine
10. Send control word to the LCD module to bring the cursor to the left corner of the module
11. Repeat the above steps from (3).

Procedure

1. Wire the circuit as shown in Fig. 22.
2. Load the program-24 into the program memory from the starting address 0000H using 89C51 microcontroller EPROM programmer.
3. Execute the program by applying +5V to the circuit
4. Observe the display of the input analog voltage on LCD module.
5. Vary the analog input voltage from 0± 5V in steps of 0.5V and record the corresponding digital data.
6. Tabulate the results in Tables 8 and 9.

Table 8

S.No.	Analog voltage DVM (Volts)	Voltage Displayed on LCD module
1	0.000	0.000
2	1.029	1.035
3	2.527	2.528
4	4.084	4.083
5	5.039	5.046
6	6.030	6.061
7	7.035	7.070
8	8.064	8.101
9	9.031	9.074
10	10.012	10.017

Table 9

S.No.	Analog voltage DVM (Volts)	Voltage Displayed on LCD module
1	0.000	0.000
2	-1.036	-1.038
3	-2.025	-2.038
4	-3.011	-3.022
5	-4.005	-4.004
6	-5.039	-5.038
7	-6.024	-6.022
8	-7.018	-7.016
9	-8.024	-8.020
10	-9.033	-9.030

Program 24 : Program for analog to digital conversion using AD1674 A/D converter

Address	Opcode	Label	Mnemonic	Operand	Comments
			;Initialization of LCD display		
0000	75 81 40	mov		sp,#40h	;initialize stack pointer ;with starting address 40h
0003	74 38	mov		a,#38h	;function set control word
0005	12 00 77	lcall		enable	;send to LCD module
0008	74 0E	mov		a,#0eh	;display ON/OFF control word
000A	12 00 77	lcall		enable	;send to LCD module
000D	74 06	mov		a,#06h	;entry mode set control word
000F	12 00 77	lcall		enable	;send to LCD module
0012	74 01	mov		a,#01h	;clear display control word
0014	12 00 77	lcall		enable	;send to LCD module
0017	74 02	mov		a,#02h	;return home control word
0019	12 00 77	lcall		enable	;send to LCD module
		rep:			
001C	12 00 A1	lcall		v_display	;call subroutine to display ;'voltage=' on LCD module
001F	12 00 39	lcall		adc	;call subroutine A/D conversion
0022	12 00 56	lcall		sub12	;call subroutine to subtract ;7FFH from the output of ADC ;for bipolar voltages
0025	12 01 05	lcall		mul_16	;call subroutine to multiply ADC ;output with 5F5EH
0028	12 01 41	lcall		hex_bcd	;call subroutine to convert ;hexadecimal number into BCD ;number

Program 24 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
002B	12 00 CA		lcall	display	;call subroutine to display ;voltage in terms of volts
002E	12 00 F8		lcall	delay	;call delay subroutine
0031	74 02		mov	a,#02h	;bring cursor to left
0033	12 00 77		lcall	enable	;corner of the display
0036	02 00 1C		ljmp	rep	;repeat the program ;A/D conversion subroutine
0039	D2 B4	adc:	setb	p3.4	;send HIGH to R/H
003B	00		nop		;small delay
003C	00		nop		
003D	00		nop		
003E	00		nop		
003F	00		nop		
0040	00		nop		
0041	00		nop		
0042	00		nop		
0043	00		nop		
0044	00		nop		
0045	00		nop		
0046	C2 B4		clr	p3.4	;send LOW to R/H to initiate ;conversion
0048	20 B5 FD		jb	p3.5, $	;poll STATUS of A/D converter ;until it goes LOW
004B	D2 B4		setb	p3.4	;send HIGH on R/H
004D	E5 80		mov	a,p0	;read low byte of ADC
004F	F8		mov	r0,a	;store it in r0
0050	E5 A0		mov	a,p2	;read higher nibble of ADC
0052	54 0F		anl	a,#0fh	;mask higher nibble
0054	F9		mov	r1,a	;store it in r1
0055	22		ret		;return to main program ;subroutine to subtract 07FFH from ADC output
0056	sub12:				
0056	E8		mov	a,r0	;read low byte of ADC from r0
0057	C3		clr	c	;clear carry flag
0058	94 FF		subb	a,#0ffh	;subtract FFH from accumulator
005A	F8		mov	r0,a	;store result in r0
005B	E9		mov	a,r1	;read higher nibble of ADC from r1
005C	94 07		subb	a,#07h	;subtract 07H from accumulator
005E	F9		mov	r1,a	;store result in r1
005F	50 13		jnc	clr_bit	;check carry flag, if C=0, ;then clear F0 flag (it is +ve Num.)
0061	D2 D5		setb	f0	;if C!=1, set F0 flag (it is -ve Num.)
					;if number is -ve, then convert it 2's complement number

Program 24 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0063	E8	mov		a,r0	;read r0)if number is -ve, then ;convert it 2's complement number
0064	F4	cpl		a	;complement accumulator
0065	F8	mov		r0,a	;store it in r0
0066	E9	mov		a,r1	;read r1
0067	F4	cpl		a	;complement accumulator
0068	F9	mov		r1,a	;store it in r1
0069	E8	mov		a,r0	;read r0
006A	24 01	add		a,#01h	;add 1 to accumulator
006C	F8	mov		r0,a	;store it in r0
006D	E9	mov		a,r1	;read r1
006E	34 00	addc		a,#00h	;add 0 along with carry
0070	F9	mov		r1,a	;store it in r1
0071	02 00 76	ljmp		skip1	;go to location skip1
0074	C2 D5	clr_bit:	clr	f0	;clear F0 flag
0076	22	skip1:		ret	;return to main program
0077		enable:			
0077	F5 90	mov		p1,a	;send control word to display
0079	C2 B4	clr		p3.2	;RS=0
007B	D2 B5	setb		p3.3	;enable=1
007D	75 F0 FF	mov		b,#0ffh	;small delay
0080	D5 F0 FD	djnz		b,$	
0083	C2 B5	clr		p3.3	;enable=0
0085	75 F0 FF	mov		b,#0ffh	;small delay
0088	D5 F0 FD	djnz		b,$	
008B	22	ret			;return to main program
008C		data_disp:			
008C	F5 90	mov		p1,a	;send character code to display
008E	D2 B4	setb		p3.2	;RS=1
0090	D2 B5	setb		p3.3	;enable=1
0092	75 F0 FF	mov		b,#0ffh	;small delay
0095	D5 F0 FD	djnz		b,$	
0098	C2 B5	clr		p3.3	;enable=0
009A	75 F0 FF	mov		b,#0ffh	;small delay
009D	D5 F0 FD	djnz		b,$	
00A0	22			ret	;return to main program
		v_display:			
00A1	74 56	mov		a,#56h	;code for 'V'
00A3	12 00 8C	lcall		data_disp	;display it on LCD module
00A6	74 6F	mov		a,#6fh	;code for 'o'
00A8	12 00 8C	lcall		data_disp	;display it on LCD module
00AB	74 6C	mov		a,#6ch	;code for 'l'
00AD	12 00 8C	lcall		data_disp	;display it on LCD module
00B0	74 74	mov		a,#74h	;code for 't'
00B2	12 00 8C	lcall		data_disp	;display it on LCD module
00B5	74 61	mov		a,#61h	;code for 'a'
00B7	12 00 8C	lcall		data_disp	;display it on LCD module

Program 24 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
00BA	74 67	mov		a,#67h	;code for 'g'
00BC	12 00 8C	lcall		data_disp	;display it on LCD module
00BF	74 65	mov		a,#65h	;code for 'e'
00C1	12 00 8C	lcall		data_disp	;display it on LCD module
00C4	74 3D	mov		a,#3dh	;code for '='
00C6	12 00 8C	lcall		data_disp	;display it on LCD module
00C9	22			ret	;return to main program
00CA		display:			
00CA	20 D5 08			jb	f0,negitive
00CD	74 2B	mov		a,#2bh	;+ve sign
00CF	12 00 8C	lcall		data_disp	
00D2	02 00 DA	ljmp		skp	
00D5	74 2D	negitive:	mov	a,#2dh	;-ve sign
00D7	12 00 8C	lcall		data_disp	
00DA		skp:			
00DA	EF	mov		a,r7	;read most significant digit
00DB	44 30	orl		a,#30h	;convert it into ASCII
00DD	12 00 8C	lcall		data_disp	;display it on LCD
00E0	74 2E	mov		a,#2eh	;load acc. with code for '.'
00E2	12 00 8C	lcall		data_disp	;display it on LCD
00E5					
00E5	EE	mov		a,r6	;read next digit
00E6	44 30	orl		a,#30h	;convert it into ASCII
00E8	12 00 8C	lcall		data_disp	;display it on LCD
00EB	ED	mov		a,r5	;read next digit
00EC	44 30	orl		a,#30h	;convert it into ASCII
00EE	12 00 8C	lcall		data_disp	;display it on LCD
00F1	EC	mov		a,r4	;read next digit
00F2	44 30	orl		a,#30h	;convert it into ASCII
0F4	12 00 8C	lcall		data_disp	;display it on LCD
00F7	22			ret	;return to main program
					;delay subroutine
00F8		delay:			
00F8	7D 07	mov		r5,#07h	
00FA	7F FF	l8:	mov	r7,#0ffh	
00FC	7E FF	l7:	mov	r6,#0ffh	
00FE	DE FE	djnz		r6,$	
0100	DF FA	djnz		r7,l7	
0102	DD F6	djnz		r5,l8	
0104	22		ret		;return to main program

```
;16X16 multiplication program
;inputs r3.r2 X r1.r0  and outputs are in r3_high byte, r2,r1,r0_low byte
;multiplicand in r3_hb,r2_lb and multiplier in r1_hb,r0_lb

                        mul_16:
```

Program 24 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0105	7A 5E	mov		r2,#5eh	
0107	7B 5F	mov		r3,#5fh	
0109					
0109	7F 00	mov		r7,#00	
010B	7E 00	mov		r6,#00	
010D					
010D	88 F0	mov		b,r0	
010F	EA	mov		a,r2	
0110	A4	mul		ab	
0111	FC	mov		r4,a	
0112	AD F0	mov		r5,b	
0114	89 F0	mov		b,r1	
0116	EA	mov		a,r2	
0117	A4	mul		ab	
0118	2D	add		a,r5	
0119	FD	mov		r5,a	
011A	E5 F0	mov		ab	
011C	3E	addc		a,r6	
011D	FE	mov		r6,a	
011E	50 01	jnc		mul_loop1	
0120	0F	inc		r7	
mul_loop1:					
0121	88 F0	mov		b,r0	
0123	EB	mov		a,r3	
0124	A4	mul		ab	
0125	2D	add		a,r5	
0126	FD	mov		r5,a	
0127	E5 F0	mov		a,b	
0129	3E	addc		a,r6	
012A	FE	mov		r6,a	
012B	50 01	jnc		mul_loop2	
012D	0F	inc		r7	
012E	mul_loop2:				
012E	89 F0	mov		b,r1	
0130	EB	mov		a,r3	
0131	A4	mul		ab	
0132	2E	add		a,r6	
0133	FE	mov		r6,a	
0134	E5 F0	mov		a,b	
0136	3F	addc		a,r7	
0137	FF	mov		r7,a	
0138	EC	mov		a,r4	
0139	F8	mov		r0,a	;l_byte
013A	ED	mov		a,r5	
013B	F9	mov		r1,a	
013C					
013C	EE	mov		a,r6	
013D	FA	mov		r2,a	
013E	EF	mov		a,r7	
013F	FB	mov		r3,a	;h_byte

Program 24 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0140	22			ret	;return to main program
					;input 32 bit HEX number is in r3_high byte, r2,r1,r0_low byte
					;output BCD number is in r0_low byte ,r1,r2,r3,r4,r5,r6,r7, 08h,09h_high byte
					hex_bcd:
0141	75 10 21	mov		10h,#21h	
0144	12 01 9F	lcall		conv1	
0147	EC	mov		a,r4	
0148	FD	mov		r5,a	
0149					
0149	75 10 21	mov		10h,#21h	
014C	12 01 9F	lcall		conv1	
014F	EC	mov		a,r4	
0150	FE	mov		r6,a	
0151	75 10 21	mov		10h,#21h	
0154	12 01 9F	lcall		conv1	
0157	EC	mov		a,r4	
0158	FF	mov		r7,a	
0159	75 10 21	mov		10h,#21h	
015C	12 01 9F	lcall		conv1	
015F	EC	mov		a,r4	
0160	F5 08	mov		08h,a	
0162	75 10 21	mov		10h,#21h	
0165	12 01 9F	lcall		conv1	
0168	EC	mov		a,r4	
0169	F5 09	mov		09h,a	
					;separation of BCDs
016B	ED	mov		a,r5	
016C	54 0F	anl		a,#0fh	
016E	F8	mov		r0,a	
016F	ED	mov		a,r5	
0170	54 F0	anl		a,#0f0h	
0172	C4	swap		a	
0173	F9	mov		r1,a	
0174	EE	mov		a,r6	
0175	54 0F	anl		a,#0fh	
0177	FA	mov		r2,a	
0178	EE	mov		a,r6	
0179	54 F0	anl		a,#0f0h	
017B	C4	swap		a	
017C	FB	mov		r3,a	
017D	EF	mov		a,r7	
017E	54 0F	anl		a,#0fh	
0180	FC	mov		r4,a	

Program 24 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0181	EF	mov		a,r7	
0182	54 F0	anl		a,#0f0h	
0184	C4	swap		a	
0185	FD	mov		r5,a	
0186	E5 08	mov		a,08h	
0188	54 0F	anl		a,#0fh	
018A	FE	mov		r6,a	
018B	E5 08	mov		a,08h	
018D	54 F0	anl		a,#0f0h	
018F	C4	swap		a	
0190	FF	mov		r7,a	
0191	E5 09	mov		a,09h	
0193	54 0F	anl		a,#0fh	
0195	F5 08	mov		08h,a	
0197	74 09	mov		a,#09h	
0199	54 F0	anl		a,#0f0h	
019B	C4	swap		a	
019C	F5 09	mov		09h,a	
019E	22	ret			
019F					
019F	C3	conv1:	clr	c	
01A0	74 00	mov		a,#00h	
01A2	FC	mov		r4,a	
01A3					
01A3	C3	conv2:	clr	c	
01A4	74 00	mov		a,#00h	
01A6	D5 10 01	djnz		10h, cont	
01A9	22	ret			
		cont:			
01AA	E8	mov		a,r0	
01AB	33	rlc		a	
01AC	F8	mov		r0,a	
01AD	E9	mov		a,r1	
01AE	33	rlc		a	
01AF	F9	mov		r1,a	
01B0	EA	mov		a,r2	
01B1	33	rlc		a	
01B2	FA	mov		r2,a	
01B3	EB	mov		a,r3	
01B4	33	rlc		a	
01B5	FB	mov		r3,a	
01B6	EC	mov		a,r4	
01B7	F5 F0	mov		b,a	
01B9	35 F0	addc		a,b	
01BB	D4	da		a	
01BC	FC	mov		r4,a	
01BD	50 E4	jnc		conv2	
01BF	E8	mov		a,r0	
01C0	34 00	addc		a,#00h	
01C2	F8	mov		r0,a	

Program 24 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
01C3	E9	mov		a,r1	
01C4	34 00	addc		a,#00h	
01C6	F9	mov		r1,a	
01C7	EA	mov		a,r2	
01C8	34 00	addc		a,#00h	
01CA	FA	mov		r2,a	
01CB	EB	mov		a,r3	
01CC	34 00	addc		a,#00h	
01CE	FB	mov		r3,a	
02 01	A3	ljmp		conv2	
01D2					
01D2		end			;program ends

```
/* program for voltage measurement using AD1674 A/D converter */

#include <stdio.h>
#include <reg51xa.h>
/* Assignment of labels for Port-2 pins */
   sbit   soc     = P3^4;
   sbit   status = P3^5;

   sbit   rs     = P3^2;
   sbit    e     = P3^3;

                int i,d,d1,d2,d3,d4;
   float v;
/* main program begins here */
   void main (void)
   {
   void display(void);
   void lcd_comand(void);
   void lcd_data(void);
   void adc_1674(void);
   void lcd_int(void);

   int c[] = {'V','o','l','t','a','g','e','='};
   lcd_int();

rep1:
   P1 = 0x02;
   lcd_comand();
```

```
   for (i = 0;i<8;i++)
   {
   P1 = c[i];
   lcd_data();
   }
   adc_1674();                //call A/D converter subroutine
   for (i=1;i<9000;i++);      //small delay

                display();
   goto rep1;
   }
/* A/D converter subroutine  */
   void adc_1674(void)
   {
   int i,lb,hb,v1;
   soc = 1;                   //Initiate conversion
   for(i=1;i<5;i++);          //small delay
   soc = 0;
   while(status == 1);        // Poll end of conversion bit until it sets
   lb = P0;                   //read low byte of ADC
   hb = P2 & 0x0f;            //read high byte of ADC
   v1 = (hb*256)+lb;
v1 = v1-0x7ff;

   if (v1<0)
   {
   F0 = 1;
   v1 = ~v1+1;
   }
   else
   F0 = 0;

    v = (v1*10.0)/4096.0;
   if (F0 == 1) // If sign bit is not set
   {
   P1 = 0x2b; // display - sign on LCD module
   lcd_data();
   }
   else
   {
   P1 = 0x2d; //else display for + sign on LCD module
   lcd_data();
   }
```

```
/* program to separate digits */
   v = v*1000;
   d = v;
   d1 = d/1000;
                d2 = d % 1000;
   d2 = d2/100;

   d3 = d % 100;
   d3 = d3/10;
   d4 = d % 10;
   }
/* display subroutine */
   void display(void)
   {
   d1 = d1+0x30;
   P1 = d1;
   lcd_data();

   P1 = 0x2e;
   lcd_data();

   d2 = d2+0x30;
   P1 = d2;
   lcd_data();

   d3 = d3+0x30;
   P1 = d3;
   lcd_data();

   d4 = d4+0x30;

   P1 = d4;
   lcd_data();
   }
 /* LCD initialization subroutine */
   void lcd_int(void)
   {
   P1 = 0x38;
   lcd_comand();

   P1 = 0x0e;
   lcd_comand();
```

```
   P1 = 0x06;
   lcd_comand();

   P1 = 0x01;
   lcd_comand();

   P1 = 0x02;
   lcd_comand();
   }

/* RS and E pulse generation subroutine for command word */
   void lcd_comand(void)
   {
   int i;
   rs = 0;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }


/* RS and E pulse generation subroutine for data  */
   void lcd_data(void)
   {
   int i;
   rs = 1;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
```

Experiment 21

AIM: 1. To interface a 16-bit analog to digital (A/D) converter AD976 and
2. To measure the analog voltage and display the same on LCD module

COMPONENTS: Microcontroller 89C51, 16-bit analog to digital converter AD976, 12 MHz quartz crystal, 1nF and 10μF capacitors, 0.1μF decoupling capacitors, 200Ω, 33kΩ, 8.2kΩ, 100Ω resistors, LCD module, bread boards and connecting wires.

EQUIPMENT: Power supply +5V/3A.

INTRODUCTION: The AD976 is 16-bit A/D converter fabricated using BiCMOS technology which operates from a single +5V supply. It is designed for microprocessor/ microcontroller control logic. It uses successive approximation as the conversion technique. The design of the AD976 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. It offers high speed, high accuracy, minimal temperature dependence, excellent long term accuracy and repeatability. It consumes less power. The features make the device ideally suited to instrumentation, process and machine control, consumer and automobile applications. The salient features of the A/D converter are:

- 16-bit resolution
- 5 μsec conversion time
- Conversion speed up to 200k samples per seconds
- Does not require external zero and full scale adjustments.
- Single + 5V power supply
- Internal 2.5V or external reference
- On chip sample and hold circuit
- On chip clock facility
- Ranges : bipolar ± 10V

Hardware: Figure 23 shows the circuit diagram of interfacing the AD976 with microcontroller. The digital output lines D0-D15 of A/D converter are connected to the port-0 and port-2 of microcontroller. The control lines R/C and BUSY of A/D

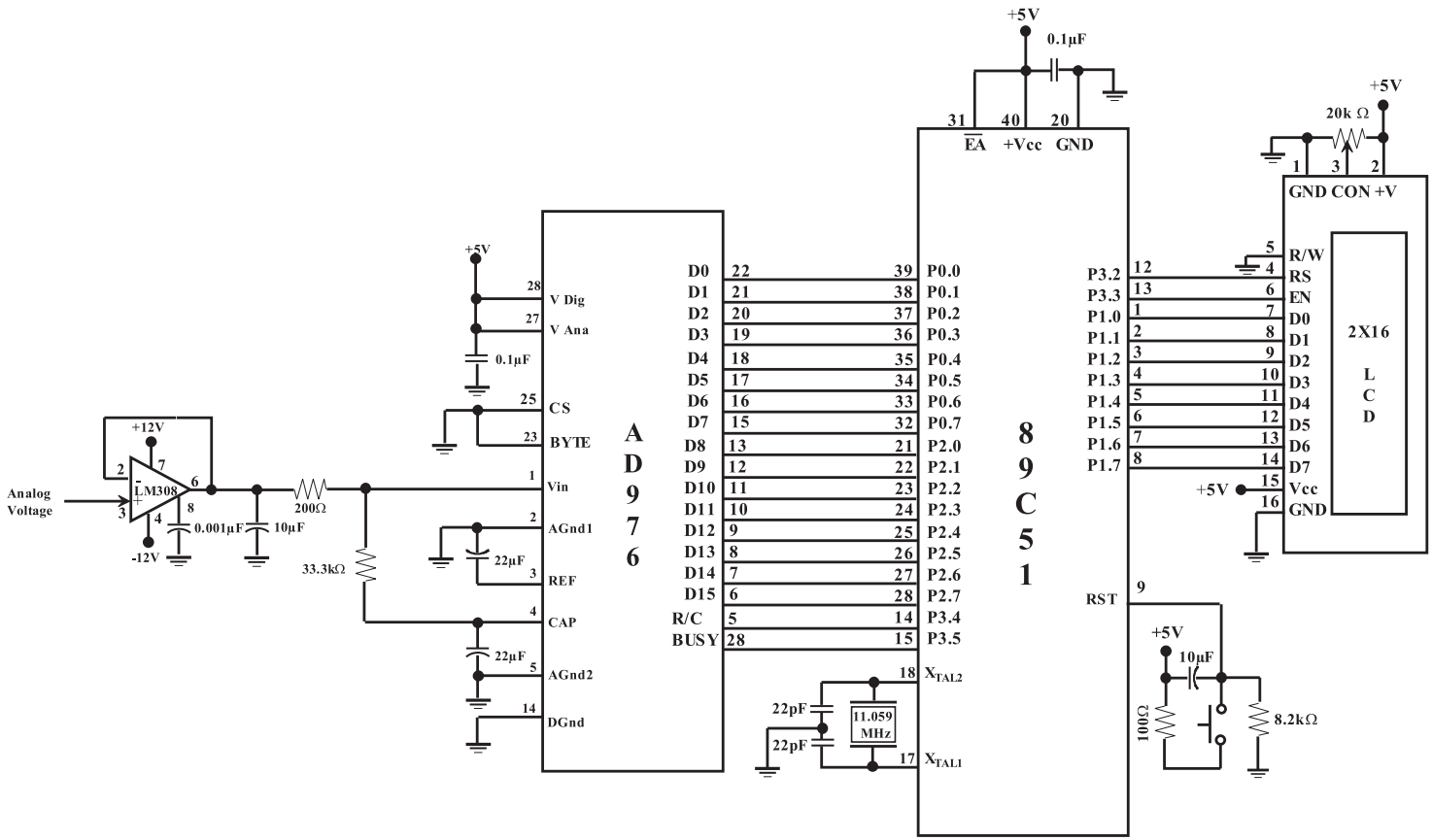


Fig. 23. The circuit diagram of interfacing the AD976 with microcontroller.

converter are connected to P3.4, P3.5 respectively as shown in Fig. 23. The circuit diagram also presents the interfacing of the display module with the microcontroller. The data bus D0-D7 of display module is connected to P1.0-P1.7 of microcontroller and the control pins RS and E are connected to the P3.2 and P3.3 pins of 89C51 respectively.

When an analog voltage is applied to the A/D converter input, the process of converting analog voltage into digital signal is started when pin R/C is made first LOW and then HIGH, this is done by P3.4 of microcontroller. To check whether the conversion is completed or not, BUSY pin is continuously monitored through P3.5. When the BUSY pin goes HIGH then conversion is completed and the digital data are available on 16 lines (D0-D15) of A/D converter which is read by microcontroller. The lower byte of digital data is read through port-0 and high byte of digital data is read through port-2. Finally after reading the 16-bit digital data, the microcontroller substitutes this digital data in the equation (2), which gives the input analog voltage directly in volts and the same is displayed on LCD module.

$$V = \frac{20^* \text{ (Digital data)}}{2^{16}} \quad \text{volts} \qquad \text{..... (1)}$$

$$V = (0.00030517)^* \text{ (Digital data)} \quad \text{volts} \qquad \text{..... (2)}$$

Software

The assembly and 'C' programs are given in program-25. The program consists of the following steps.

a. Initialize the stack pointer with the starting address 30H
b. Initialize LCD module
c. Call subroutine to display 'Voltage=' on LCD module
d. Call subroutine to convert analog voltage into digital data
e. Call subroutine to multiply the digital data with 30517 (7735H) to convert the digital data into analog voltage.
f. Call subroutine to convert binary number into BCD number
g. Call subroutine to display analog voltage on LCD module
h. Call delay subroutine
i. Send control word to the LCD module to bring the cursor to left corner of the module
j. Repeat the above steps from (c).

Procedure

1. Wire the circuit as shown in Fig. 23.
2. Load the program-25 into the program memory from the starting address 0000H using 89C51 microcontroller programmer.
3. Execute the program by applying +5V to the circuit

4. Observe the displaying of the analog input voltage applied to the A/D converter on the LCD module.
5. Vary the analog input voltage from 0±10V in steps of 0.5V and record the corresponding digital data.
6. Tabulate the results in Table 10 and 11.

Table 10

S.No.	Analog voltage DVM (Volts)	Voltage Displayed on LCD module
1	0.0000	0.0000
2	0.5009	0.5001
3	1.0009	1.0006
4	1.5028	1.5011
5	2.0002	2.0012
6	2.5025	2.5028
7	3.0005	3.0012
8	3.5011	3.5026
9	4.0021	4.0032
10	4.5009	4.5015
11	5.0006	5.0013
12	5.5002	5.5009
13	6.0016	6.0018
14	6.5003	6.5007
15	7.0004	7.0006
16	7.5014	7.5019
17	8.0022	8.0027
18	8.5001	8.5003
19	9.0026	9.0030
20	9.5005	9.5009

Table 11

S.No.	Analog voltage DVM (Volts)	Voltage Displayed on LCD module
1	0.0000	0.0000
2	-0.5090	-0.5081
3	-1.0009	-1.0012
4	-1.5026	-1.5041
5	-2.0028	-2.0036
6	-2.5002	-2.5012
7	-3.0011	-3.0021
8	-3.5023	-3.5035
9	-4.0006	-4.0013
10	-4.5015	-4.5026
11	-5.0021	-5.0035
12	-5.5017	-5.5021
13	-6.0008	-6.0019
14	-6.5000	-6.5008
15	-7.0014	-7.0023
16	-7.5008	-7.5013
17	-8.0020	-8.0029
18	-8.5002	-8.5009
19	-9.0025	-9.5018
20	-9.5022	-9.5024
	-9.030	

Program 25: Program for analog to digital conversion using 16 bit A/D converter AD976

Address	Opcode	Label	Mnemonic	Operand	Comments
0000	12 00 EA	lcall		delay	;Initialization of LCD display
0003	75 81 40	mov		sp,#40h	;Initialize stack pointer
					;with starting address 40h
0006	74 38	mov		a,#38h	;Initialize LCD module
0008	12 00 62	lcall		enable	
000B	74 0E	mov		a,#0eh	
000D	12 00 62	lcall		enable	
0010	74 06	mov		a,#06h	
0012	12 00 62	lcall		enable	
0015	74 01	mov		a,#01h	
0017	12 00 62	lcall		enable	
001A	74 02	mov		a,#02h	
001C	12 00 62	lcall		enable	
		rep:			
001F	12 00 8C	lcall		v_display	;call display subroutine to
					;display with 'Voltage=' on
					;LCD module
0022	12 00 39	lcall		adc	;call A/D converter subroutine
					;to convert analog voltage
					;into digital data
0025	12 00 F7	lcall		mul_16	;call 16-bit multiplication
					;subroutine to multiply
					;digital data with 30517
0028	12 01 33	lcall		hex_bcd	;call subroutine to convert
					;32-bit binary into BCD
002B	12 00 B5	lcall		display	;call display subroutine
					;to display voltage on LCD
					;module
002E	12 00 EA	lcall		delay	;call delay subroutine
0031	74 02	mov		a,#02h	;Bring cursor to left corner
0033	12 00 62	lcall		enable	;of LCD module
0036	02 00 1F	ljmp		rep	;repeat above procedure to
					;display voltage continuously
					;LCD module
					;A/D conversion program
0039	C2 B4	adc:	clr	p3.4	;send LOW to start of
					;conversion
003B	00	nop			
003C	00	nop			
003D	00	nop			
003E	D2 B4	setb		p3.4	;send HIGH to start of conversion
					;to begin conversion
0040	30 B5 FD	jnb		p3.5,$	;Poll end of conversion signal
					;until it goes HIGH
0043	E5 80	mov		a,p0	;read low byte of A/D converter
0045	F8	mov		r0,a	;store in register r0
0046	E5 A0	mov		a,p2	;read high byte of A/D converter
0048	F9	mov		r1,a	;store in register r1
0049	30 E7 13	jnb		acc.7,clr_bit	;check whether input is -ve
					;or +ve

Program 25 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
004C	D2 D5	setb		f0	;set user flag if data is -ve ;if the data is negative, it will be in two's ;complement form. Hence, it is to be converted ;into normal form
004E	E8	mov		a,r0	
004F	F4	cpl		a	
0050	F8	mov		r0,a	
0051	E9	mov		a,r1	
0052	F4	cpl		a	
0053	F9	mov		r1,a	
0054	E8	mov		a,r0	
0055	24 01	add		a,#01h	
0057	F8	mov		r0,a	
0058	E9	mov		a,r1	
0059	34 00	addc		a,#00h	
005B	F9	mov		r1,a	
005C	02 00 61	ljmp		skip1	
005F	C2 D5	clr_bit:	clr	f0	;If data is +ve then clear ;user flag
0061	22	skip1:		ret	
0062	enable:				
0062	F5 90	mov		p1,a	
0064	C2 B2	clr		p3.2	;rs=0
0066	D2 B3	setb		p3.3	;rs=0,e=1
0068	75 F0 FF	mov		b,#0ffh	
006B	D5 F0 FD	djnz		b,$	
006E	C2 B3	clr		p3.3	
0070	75 F0 FF	mov		b,#0ffh	
0073	D5 F0 FD	djnz		b,$	
0076	22	ret			
	data_disp:				
0077	F5 90	mov		p1,a	
0079	D2 B2	setb		p3.2	;rs=1
007B	D2 B3	setb		p3.3	;rs=1,e=1
007D	75 F0 FF	mov		b,#0ffh	
0080	D5 F0 FD	djnz		b,$	
0083	C2 B3	clr		p3.3	
0085	75 F0 FF	mov		b,#0ffh	
0088	D5 F0 FD	djnz		b,$	
008B	22	ret			
		v_display:			
008C	74 56	mov		a,#56h	
008E	12 00 77	lcall		data_disp	
0091	74 6F	mov		a,#6fh	
0093	12 00 77	lcall		data_disp	
0096	74 6C	mov		a,#6ch	
0098	12 00 77	lcall		data_disp	
009B	74 74	mov		a,#74h	
009D	12 00 77	lcall		data_disp	

Program 25 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
00A0	74 61	mov		a,#61h	
00A2	12 00 77	lcall		data_disp	
00A5	74 67	mov		a,#67h	
00A7	12 00 77	lcall		data_disp	
00AA	74 65	mov		a,#65h	
00AC	12 00 77	lcall		data_disp	
00AF	74 3D	mov		a,#3dh	
00B1	12 00 77	lcall		data_disp	
00B4	22	ret			
00B5	display:				
00B5	20 D5 08	jb		f0,negitive	
00B8	74 2B	mov		a,#2bh	;+ve sign
00BA	12 00 77	lcall		data_disp	
00BD	02 00 C5	ljmp		skip2	
00C0	74 2D	negitive:	mov	a,#2dh	;-ve sign
00C2	12 00 77	lcall		data_disp	
00C5					
00C5	E5 08	skip2:	mov	a,08h	
00C7	44 30	orl		a,#30h	
00C9	12 00 77	lcall		data_disp	
00CC	74 2E	mov		a,#2eh	
00CE	12 00 77	lcall		data_disp	
00D1					
00D1	EF	mov		a,r7	
00D2	44 30	orl		a,#30h	
00D4	12 00 77	lcall		data_disp	
00D7	EE	mov		a,r6	
00D8	44 30	orl		a,#30h	
00DA	12 00 77	lcall		data_disp	
00DD	ED	mov		a,r5	
00DE	44 30	orl		a,#30h	
00E0	12 00 77	lcall		data_disp	
00E3	EC	mov		a,r4	
00E4	44 30	orl		a,#30h	
00E6	12 00 77	lcall		data_disp	
00E9	22	ret			
00EA	delay:				
00EA	7D 08	mov		r5,#08h	
00EC	7F FF	l8:	mov	r7,#0ffh	
00EE	7E FF	l7:	mov	r6,#0ffh	
00F0	DE FE		djnz	r6,$	
00F2	DF FA		djnz	r7,l7	
00F4	DD F6		djnz	r5,l8	
00F6	22		ret		
					;16X16 multiplication program ;inputs r3.r2 X r1.r0 ;outputs r3_high byte, r2_low byte mul_16:

Program 25 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
					;multiplicand in r3_hb,r2_lb
					;multiplier in r1_hb,r0_lb
00F7	7A 35	mov		r2,#35h	
00F9	7B 77	mov		r3,#77h	
00FB					
00FB	7F 00	mov		r7,#00	
00FD	7E 00	mov	r6,#00		
00FF					
00FF	88 F0	mov		b,r0	
0101	EA	mov		a,r2	
0102	A4	mul		ab	
0103	FC	mov	r4,a		
0104	AD F0	mov		r5,b	
0106	89 F0	mov		b,r1	
0108	EA	mov		a,r2	
0109	A4	mul		ab	
010A	2D	add		a,r5	
010B	FD	mov		r5,a	
010C	E5 F0	mov		ab	
010E	3E	addc		a,r6	
010F	FE	mov		r6,a	
0110	50 01	jnc		mul_loop1	
0112	0F	inc		r7	
	mul_loop1:				
0113	88 F0	mov		b,r0	
0115	EB	mov		a,r3	
0116	A4	mul		ab	
0117	2D	add		a,r5	
0118	FD	mov		r5,a	
0119	E5 F0	mov		ab	
011B	3E	addc		a,r6	
011C	FE	mov		r6,a	
011D	50 01	jnc	mul_loop2	0F	inc r7
	mul_loop2:		011F		
0120	89 F0	mov		b,r1	
0122	EB	mov		a,r3	
0123	A4	mul		ab	
0124	2E	add		a,r6	
0125	FE	mov		r6,a	
0126	E5 F0	mov		a,b	
0128	3F	addc		a,r7	
0129	FF	mov		r7,a	
012A	EC	mov		a,r4	
012B	F8	mov		r0,a ;l_byte	
012C	ED	mov		a,r5	
012D	F9	mov		r1,a	
012E					
012E	EE	mov		a,r6	
012F	FA	mov		r2,a	
0130	EF	mov		a,r7	

Program 25 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0131	FB	mov		r3,a	;h_byte
0132	22	ret			
					;input 32 bit hex number is in r3_hb,r2,r1,r0_lb
					;output BCD number is in r0_lb,r1,r2,r3,r4,r5,r6,r7,08h, 09h_hb
0133		hex_bcd:			
0133	75 10 21	mov		10h,#21h	
0136	12 01 91	lcall		conv1	
0139	EC	mov		a,r4	
013A	FD	mov		r5,a	
013B	75 10 21	mov		10h,#21h	
013E	12 01 91	lcall		conv1	
0141	EC	mov		a,r4	
0142	FE	mov		r6,a	
0143	75 10 21	mov		10h,#21h	
0146	12 01 91	lcall		conv1	
0149	EC	mov		a,r4	
014A	FF	mov		r7,a	
014B	75 10 21	mov		10h,#21h	
014E	12 01 91	lcall		conv1	
0151	EC	mov		a,r4	
0152	F5 08	mov		08h,a	
0154	75 10 21	mov		10h,#21h	
0157	12 01 91	lcall		conv1	
015A	EC	mov		a,r4	
015B	F5 09	mov		09h,a	
					;seperation of BCDs
015D	ED	mov		a,r5	
015E	54 0F	anl		a,#0fh	
0160	F8	mov		r0,a	
0161	ED	mov		a,r5	
0162	54 F0	anl		a,#0f0h	
0164	C4	swap		a	
0165	F9	mov		r1,a	
0166	EE	mov		a,r6	
0167	54 0F	anl		a,#0fh	
0169	FA	mov		r2,a	
016A	EE	mov		a,r6	
016B	54 F0	anl		a,#0f0h	
016D	C4	swap		a	
016E	FB	mov		r3,a	
016F					
016F	EF	mov		a,r7	
0170	54 0F	anl		a,#0fh	
0172	FC	mov		r4,a	
0173	EF	mov		a,r7	
0174	54 F0	anl		a,#0f0h	
0176	C4	swap		a	
0177	FD	mov		r5,a	
0178	E5 08	mov		a,08h	
017A	54 0F	anl		a,#0fh	
017C	FE	mov		r6,a	

Program 25 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
017D	E5 08	mov		a,08h	
017F	54 F0	anl		a,#0f0h	
0181	C4	swap		a	
0182	FF	mov		r7,a	
0183	E5 09	mov		a,09h	
0185	54 0F	anl		a,#0fh	
0187	F5 08	mov		08h,a	
0189	74 09	mov		a,#09h	
018B	54 F0	anl		a,#0f0h	
018D	C4	swap		a	
018E	F5 09	mov		09h,a	
0190	22	ret			
0191	C3	conv1:	clr	c	
0192	74 00	mov		a,#00h	
0194	FC	mov		r4,a	
0195					
0195	C3	conv2:	clr	c	
0196	74 00	mov		a,#00h	
0198	D5 10 01	djnz		10h, cont	
019B	22	ret			
019C	cont:				
019C	E8	mov		a,r0	
019D	33	rlc		a	
019E	F8	mov		r0,a	
019F	E9	mov		a,r1	
01A0	33	rlc		a	
01A1	F9	mov		r1,a	
01A2	EA	mov		a,r2	
01A3	33	rlc		a	
01A4	FA	mov		r2,a	
01A5	EB	mov		a,r3	
01A6	33	rlc		a	
01A7	FB	mov		r3,a	
01A8	EC	mov		a,r4	
01A9	F5 F0	mov		b,a	
01AB	35 F0	addc		a,b	
01AD	D4	da		a	
01AE	FC	mov		r4,a	
01AF	50 E4	jnc		conv2	
01B1	E8	mov		a,r0	
01B2	34 00	addc		a,#00h	
01B4	F8	mov		r0,a	
01B5	E9	mov		a,r1	
01B6	34 00	addc		a,#00h	
01B8	F9	mov		r1,a	
01B9	EA	mov		a,r2	
01BA	34 00	addc		a,#00h	
01BC	FA	mov		r2,a	
01BD	EB	mov		a,r3	
01BE	34 00	addc		a,#00h	
01C0	FB	mov		r3,a	
01C1	02 01 95	ljmp		conv2	
		end			

Experiment 22

AIM: 1. To interface a 12-bit A/D converter ICL7109 with microcontroller and
2. To measure the analog voltage and display the same on LCD module.

COMPONENTS: 89C51, A/D converter ICL7109, 2-5.1V zener diodes, 100kΩ pot, 0.15μF, 0.33μF, 0.01μF metalized polyester capacitors (one each), 1.0μF tantalum capacitor,10μF capacitor, 20kΩ, 5kΩ multi-turn pot (one each), 1MΩ, 100kΩ, 22kΩ, 8.2kΩ,100Ω, resistors, 3.5795 MHz, 12.0 MHz crystals, switch, bread board and connecting wires.

EQUIPMENT: Power supply + 5V/2A, ± 12V/100mA

INTRODUCTION: The Intersil ICL7109 is a 12-bit A/D converter. The device offers high accuracy by lowering rollover error less than 1 count and zero reading drift to less than 1μV/^{O}C. In many data acquisition systems the ICL7109 is attractive because of low cost, low power consumption and very low input bias current.

Hardware: Figure 24 shows interfacing of 12-bit A/D converter ICL7109 with the microcontroller. The +5V and –5V voltages for A/D converter are generated using 5.1V zener diodes. A metalized polyester capacitor 'C' (0.15μF) and resistor 'R' (100kΩ) acts as RC for integrator. The 1M? and 0.01μF capacitor at the input pins IN HI and IN LO are used to prevent any noise picked up on the input leads from entering the IC. A 3.5795MHz crystal is connected in between OSC-IN and OSC-OUT out pins of A/D converter to provide clock for A/D conversion. The A/D converter has 14 tri-state outputs: 12 bit data bits, 1 polarity and 1 over range bit. These bits are enabled by connecting CE/LOAD, LBEN and HBEN control signals to ground. Here RUN/ HOLD input is tied +5V for continuous conversions with a fixed length of 8192 clock cycles per conversion. At the end of the conversion cycle the STATUS output goes LOW, this signal is used for knowing whether conversion is completed or not for reading the digital data from an A/D converter. An input of +1V will generate the same 12-bit data output as an input of -1V. However, the polarity pin will be at logic HIGH for positive inputs and at logic LOW for negative. The on-chip reference

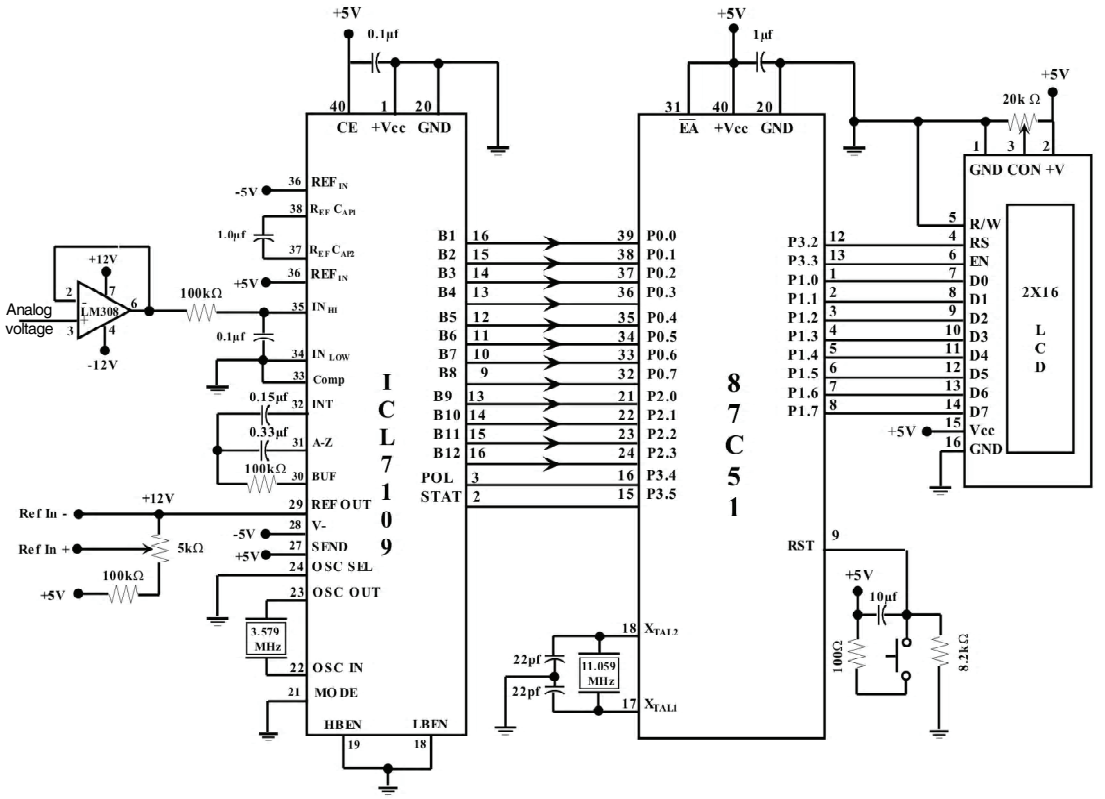


Fig. 24. Interfacing of 12-bit A/D converter ICL7109 with the microcontroller.

voltage (2V) derived from REF OUT is applied to the differential reference. Hence, in the present experiment the range of A/D converter is ±2.0V.

The A/D converter is interfaced through port-0 and port-2 of microcontroller. The pins POL and STATUS of A/D converter are connected to the P3.6 and P3.5 of microcontroller respectively. The 12 digital outputs of A/D are connected to port-0 and four lines (P2.0 – P2.3) of port-2 of microcontroller. The circuit diagram also presents the interfacing of the LCD display module with the microcontroller. The data bus D0-D7 of display module is connected to P1.0-P1.7 of microcontroller and the control pins RS and E are connected to the P3.2 and P3.3 pins of 89C51 respectively.

After applying an analog voltage to the A/D converter input, STATUS pin is continuously monitored through P3.5 to check whether the conversion is completed or not. When the STATUS pin goes low then conversion is completed and the digital data are available on 12 lines (D0-D11) of A/D converter which is read by microcontroller. The lower byte of digital data is read through port-0 and high nibble of digital data is read through port-2 (P2.0-P2.3). Finally after reading digital data, the microcontroller substitutes this digital data in equation (3), which gives the input analog voltage directly in volts and the same is displayed on LCD module.

$$V = \frac{\text{Vref} * (\text{Digital data})}{4096} \quad \text{volts} \quad \text{..... (1)}$$

$$V = (2/4096)*\text{digital data} \quad \text{volts} \quad \text{..... (2)}$$

$$V = 0.000488* \text{ digital data} \quad \text{volts} \quad \text{..... (3)}$$

Software Description : The assembly language program and 'C' programs are given in program-26. The software first initializes stack pointer and LCD module. After initialization the program then calls the A/D conversion subroutine to read the analog voltage through A/D converter ICL7109. This digital data from A/D converter is multiplied with 488 (01E8H) get analog voltage applied and same will be displayed on the LCD module.

a. Initialize the stack pointer
b. Initialize the LCD module
c. Call subroutine to display 'Voltage=' on LCD module
d. Call subroutine to convert analog voltage into digital data
e. Call subroutine to multiply the digital data with 488(01E8H) to convert the digital data into analog voltage.
f. Call subroutine to convert binary number into BCD number
g. Call subroutine to display voltage on LCD module
h. Call delay subroutine
i. Send control word to the LCD module to bring the cursor to the left corner of the module
j. Repeat the steps above from (c).

Procedure

1. Wire the circuit as shown in Fig. 24.
2. Load the program-26 into program memory from the starting address 0000h of through 89C51 programmer.
3. Execute the program by applying 5V to the circuit. Observe the continuous display of the analog input voltage LCD module.
4. Vary the analog input voltage from 0±2V in steps of 0.1V and record the corresponding digital data.
5. Tabulate the results in Table 12 and 13.

Table 12

S.No.	Analog voltage DVM (Volts)	Voltage Displayed on LCD module
1	0.000	0.000
2	0.010	0.011
3	0.123	0.124
4	0.256	0.256
5	0.468	0.468
6	0.752	0.751
7	1.018	1.018
8	1.399	1.398
9	1.815	1.815
10	2.000	2.000

Table 13

S.No.	Analog voltage DVM (Volts)	Voltage Displayed on LCD module
1	0.000	0.000
2	-0.200	-0.198
3	-0.327	-0.328
4	-0.589	-0.589
5	-0.821	-0.820
6	-1.154	-1.154
7	-1.233	-1.234
8	-1.455	-1.455
9	-1.789	-1.788
10	-2.000	-1.999

Program 26 : Program for conversion of analog voltage into digital using ICL7109

Address	Opcode	Label	Mnemonic	Operand	Comments
		org	0000h		;starting address of the program
					;Initialization of LCD display
0000	75 81 40	mov		sp,#40h	;Initialize stack pointer
					;with starting address 40h
0003	74 38	mov		a,#38h	;function set control word
0005	12 00 40	lcall		enable	;send to LCD module
0008	74 0E	mov		a,#0eh	;display ON/OFF control word
000A	12 00 40	lcall		enable	;send to LCD module
000D	74 06	mov		a,#06h	;entry mode set control word
000F	12 00 40	lcall		enable	;send to LCD module
0012	74 01	mov		a,#01h	;clear display control word
0014	12 00 40	lcall		enable	;send to LCD module
0017	74 02	mov		a,#02h	;return home control word
0019	12 00 40	lcall		enable	;send to LCD module

Program 26 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
		rep:			
001C	12 00 6A	lcall		v_display	;call subroutine to display ;'Voltage=' on LCD module
001F	12 00 33	lcall		adc	;call subroutine A/D conversion
0022	12 00 CE	lcall		mul_1	;call subroutine to multiply ADC ;data with 488(1E8H)
0025	12 01 0C	lcall		hex_bcd	;call subroutine to convert ;hexadecimal number into BCD ;number
0028	12 00 93	lcall		display	;call subroutine to display ;voltage in terms of V
002B	74 02	mov		a,#02h	;bring cursor to left
002D	12 00 40	lcall		enable	;corner of the display
0030	02 00 1C	ljmp		rep	;repeat the program
		adc:			
0033	20 B5 FD	jb		p3.5,$	;poll STATUS until it goes LOW
0036	E5 80	mov		a,p0	;read D0-D7 of A/D converter
0038	F8	mov		r0,a	;store it in r0
0039	E5 A0	mov		a,p2	;read D8-D11 of A/D converter
003B	54 0F	anl		a,#0fh	;mask higher nibble
003D	52 01	anl		r1,a	;store it in r1
003F	22	ret			;return to main program
		enable:			
0040	F5 90	mov		p1,a	;send control word to display
0042	C2 B2	clr		p3.2	;RS=0
0044	D2 B3	setb		p3.3	;enable=1
0046	75 F0 FF	mov		b,#0ffh	;small delay
0049	D5 F0 FD	djnz		b,$	
004C	C2 B3	clr		p3.3	;enable=0
004E	75 F0 FF	mov		b,#0ffh	;small delay
0051	D5 F0 FD	djnz		b,$	
0054	22	ret			;return to main program
		data_disp:			
0055	F5 90	mov		p1,a	;send character code to display
0057	D2 B2	setb		p3.2	;RS=1
0059	D2 B3	setb		p3.3	;enable=1
005B	75 F0 FF	mov		b,#0ffh	;small delay
005E	D5 F0 FD	djnz		b,$	
0061	C2 B3	clr		p3.3	;enable=0
0063	75 F0 FF	mov		b,#0ffh	;small delay
0066	D5 F0 FD	djnz		b,$	
0069	22	ret			;return to main program
		v_display:			
006A	74 56	mov		a,#56h	;code for 'V'
006C	12 00 55	lcall		data_disp	;display it on LCD module
006F	74 6F	mov		a,#6fh	;code for 'o'
0071	12 00 55	lcall		data_disp	;display it on LCD module
0074	74 6C	mov		a,#6ch	;code for 'l'
0076	12 00 55	lcall		data_disp	;display it on LCD module
0079	74 74	mov		a,#74h	;code for 't'

Program 26 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
007B	12 00 55	lcall		data_disp	;display it on LCD module
007E	74 61	mov		a,#61h	;code for 'a'
0080	12 00 55	lcall		data_disp	;display it on LCD module
0083	74 67	mov		a,#67h	;code for 'g'
0085	12 00 55	lcall		data_disp	;display it on LCD module
0088	74 65	mov		a,#65h	;code for 'e'
008A	12 00 55	lcall		data_disp	;display it on LCD module
008D	74 3D	mov		a,#3dh	;code for '='
008F	12 00 55	lcall		data_disp	;display it on LCD module
0092	22	ret			;return to main program
		display:			
0093	30 B4 08	jnb		p3.4, negative	;if polarity bit is not set then ;then voltage is –ve else +ve
0096	74 2B	mov		a,#2bh	;code for +ve sign
0098	12 00 55	lcall		data_disp	
009B	02 00 A3	ljmp		skp	
009E	74 2D	negative: mov		a,#2dh	;code for -ve sign
00A0	12 00 55	lcall		data_disp	
		skp:			
00A3	E9	mov		a,r1	;read most significant digit
00A4	44 30	orl		a,#30h	;convert it into ASCII
00A6	12 00 55	lcall		data_disp	;display it on LCD
00A9	74 2E	mov		a,#2eh	;load acc. with code for '.'
00AB	12 00 55	lcall		data_disp	;display it on LCD
00AE	EA	mov		a,r2	;read next digit
00AF	44 30	orl		a,#30h	;convert it into ASCII
00B1	12 00 55	lcall		data_disp	;display it on LCD
00B4	EB	mov		a,r3	;read next digit
00B5	44 30	orl		a,#30h	;convert it into ASCII
00B7	12 00 55	lcall		data_disp	;display it on LCD
00BA	EC	mov		a,r4	;read next digit
00BB	44 30	orl		a,#30h	;convert it into ASCII
00BD	12 00 55	lcall		data_disp	;display it on LCD
00C0	22	ret			;return to main program
00C1	delay:				
00C1	7D 04	mov		r5,#04h	
00C3	7F FF	l8:	mov	r7,#0ffh	
00C5	7E FF	l7:	mov	r6,#0ffh	
00C7	DE FE	djnz		r6,$	
00C9	DF FA	djnz		r7,l7	
00CB	DD F6	djnz		r5,l8	
00CD	22	ret			
					;16X16 multiplication program ;inputs r3.r2 X r1.r0 and outputs will be in r3,r2_lb ;multiplicand in r3_hb,r2_lb and

Program 26 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
					multiplier in r1_hb,r0_lb
	mul_16:				
00CE	7A E8		mov	r2,#0e8h	
00D0	7B 01		mov	r3,#01h	
00D2	7F 00		mov	r7,#00	
00D4	7E 00		mov	r6,#00	
00D6	88 F0		mov	b,r0	
00D8	EA		mov	a,r2	
00D9	A4		mul	ab	
00DA	FC		mov	r4,a	
00DB	AD F0		mov	r5,b	
00DD	89 F0		mov	b,r1	
00DF	EA		mov	a,r2	
00E0	A4		mul	ab	
00E1	2D		add	a,r5	
00E2	FD		mov	r5,a	
00E3	E5 F0		mov	a,b	
00E5	3E		addc	a,r6	
00E6	FE		mov	r6,a	
00E7	50 01		jnc	mul_loop1	
00E9	0F		inc	r7	
	mul_loop1:				
00EA	88 F0		mov	b,r0	
00EC	EB		mov	a,r3	
00ED	A4		mul	ab	
00EE	2D		add	a,r5	
00EF	FD		mov	r5,a	
00F0	E5 F0		mov	a,b	
00F2	3E		addc	a,r6	
00F3	FE		mov	r6,a	
00F4	50 01		jnc	mul_loop2	
00F6	0F		inc	r7	
	mul_loop2:				
00F7	89 F0		mov	b,r1	
00F9	EB		mov	a,r3	
00FA	A4		mul	ab	
00FB	2E		add	a,r6	
00FC	FE		mov	r6,a	
00FD	E5 F0		mov	a,b	
00FF	3F		addc	a,r7	
0100	FF		mov	r7,a	
0101	EC		mov	a,r4	
0102	FA		mov	r2,a ;l_byte	
0103	ED		mov	a,r5	
0104	FB		mov	r3,a ;h_byte	
0105	EA		mov	a,r2	
0106	F8		mov	r0,a	
0107	EB		mov	a,r3	
0108	F9		mov	r1,a	
0109	EE		mov	a,r6	

Program 26 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
010A	FA	mov		r2,a	
010B	22	ret			
					;input 24 bit hex number is in r2_hb,r1,r0_lb
					;output bcd number is in r0_hb,r1,r2,r3,r4,r5,r6,r7_lb
	hex_bcd:				
010C	7F 19	mov		r7,#19h	
010E	12 01 6D	lcall		conv1	
0111	EB	mov		a,r3	
0112	FC	mov		r4,a	
0113	7F 19	mov		r7,#19h	
0115	12 01 6D	lcall		conv1	
0118	EB	mov		a,r3	
0119	FD	mov		r5,a	
011A	7F 19	mov		r7,#19h	
011C	12 01 6D	lcall		conv1	
011F	EB	mov		a,r3	
0120	FE	mov		r6,a	
0121	7F 19	mov		r7,#19h	
0123	12 01 6D	lcall		conv1	
0126	EB	mov		a,r3	
0127	FF	mov		r7,a	
0128	EC	mov		a,r4	
0129	54 0F	anl		a,#0fh	
012B	F8	mov		r0,a	
012C	EC	mov		a,r4	
012D	54 F0	anl		a,#0f0h	
012F	C4	swap		a	
0130	F9	mov		r1,a	
0131	ED	mov		a,r5	
0132	54 0F	anl		a,#0fh	
0134	FA	mov		r2,a	
0135	ED	mov		a,r5	
0136	54 F0	anl		a,#0f0h	
0138	C4	swap		a	
0139	FB	mov		r3,a	
013A	EE	mov		a,r6	
013B	54 0F	anl		a,#0fh	
013D	FC	mov		r4,a	
013E	EE	mov		a,r6	
013F	54 F0	anl		a,#0f0h	
0141	C4	swap		a	
0142	FD	mov		r5,a	
0143	EF	mov		a,r7	
0144	54 0F	anl		a,#0fh	
0146	FE	mov		r6,a	
0147	EF	mov		a,r7	
0148	54 F0	anl		a,#0f0h	
014A	C4	swap		a	
014B	FF	mov		r7,a	

Program 26 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
014C	EF		mov	a,r7	
014D	F5 08		mov	08h,a	
014F	EE		mov	a,r6	
0150	F5 09		mov	09h,a	
0152	ED		mov	a,r5	
0153	F5 0A		mov	0ah,a	
0155	EC		mov	a,r4	
0156	F5 0B		mov	0bh,a	
0158	E8		mov	a,r0	
0159	FF		mov	r7,a	
015A	E9		mov	a,r1	
015B	FE		mov	r6,a	
015C	EA		mov	a,r2	
015D	FD		mov	r5,a	
015E	EB		mov	a,r3	
015F	FC		mov	r4,a	
0160	E5 08		mov	a,08h	
0162	F8		mov	r0,a	
0163	E5 09		mov	a,09h	
0165	F9		mov	r1,a	
0166	E5 0A		mov	a,0ah	
0168	FA		mov	r2,a	
0169	E5 0B		mov	a,0bh	
016B	FB		mov	r3,a	
016C	22		ret		
016D	C3	conv1:	clr	c	
016E	74 00		mov	a,#00h	
0170	FB		mov	r3,a	
0171	C3	conv2:	clr	c	
0172	74 00		mov	a,#00h	
0174	DF 01		djnz	r7, cont	
0176	22		ret		
	cont:				
0177	E8		mov	a,r0	
0178	33		rlc	a	
0179	F8		mov	r0,a	
017A	E9		mov	a,r1	
017B	33		rlc	a	
017C	F9		mov	r1,a	
017D	EA		mov	a,r2	
017E	33		rlc	a	
017F	FA		mov	r2,a	
0180	EB		mov	a,r3	
0181	F5 F0		mov	b,a	
0183	35 F0		addc	a,b	
0185	D4		da	a	
0186	FB		mov	r3,a	
0187	50 E8		jnc	conv2	
0189	E8		mov	a,r0	

Program 26 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
018A	34 00	addc		a,#00h	
018C	F8	mov		r0,a	
018D	E9	mov		a,r1	
018E	34 00	addc		a,#00h	
0190	F9	mov		r1,a	
0191	EA	mov		a,r2	
192	34 00	addc		a,#00h	
0194	FA	mov		r2,a	
0195	02 01 71	ljmp		conv2	
		end			

```
/* program for voltage measurement using ICL7109 A/D converter */

#include <stdio.h>
#include <reg51.h>

   sbit   sign = P3^4;   //POLARITY
   sbit   eoc  = P3^5;   //STATUS

   sbit   rs = P3^2;
   sbit    e = P3^3;
        int d,d1,d2,d3,d4;

   void main (void)

   {
   void adc_7109(void);
   void lcd_comand(void);
   void lcd_data(void);
   void lcd_int(void);
   int i;

   int v[] = {'V','o','l','t','a','g','e','='};

   lcd_int();
rep1:

   P1 = 0x02;
   lcd_comand();

   for (i = 0;i<8;i++)
   {
```

```
   P1 = v[i];
   lcd_data();
   }
   adc_7109();
   for (i=1;i<9000;i++);   // small delay
        display();
   goto rep1;

   }
/* function for analog to digital conversion */
   void adc_7109(void)

   {
   int lb,hb;
   float v;

   if (sign == 1)  // If sign bit is not set
   {
   P1 = 0x2b; // display '-' sign on LCD module
   lcd_data();
   }
   else
   {
   P1 = 0x2d; //else display for '+' sign on LCD module
   lcd_data();
   }
   /* Poll end of conversion bit until it sets */
   while(eoc!=1);
   lb = P0;         //read low byte of ADC
   hb = P2 & 0x0f;
    v = (hb*256)+lb;  //read high byte of ADC
    v = (0.4*v)/4096; //convert it into voltage
    v = v*1000;    //separation of digits
    d = v;
   d1 = d/1000;

   d2 = d % 1000;
   d2 = d2/100;

   d3 = d % 100;
   d3 = d3/10;
   d4 = d % 10;
   }
```

```
/* function for displaying analog voltage on LCD module */
   void display(void)
   {
   d1 = d1+0x30;
   P1 = d1;
   lcd_data();

   P1 = 0x2e;
   lcd_data();

   d2 = d2+0x30;
   P1 = d2;
   lcd_data();

   d3 = d3+0x30;
   P1 = d3;
   lcd_data();

   d4 = d4+0x30;
   P1 = d4;
   lcd_data();
   }
 /* function for initialization of LCD module */
   void lcd_int(void)
   {
   P1 = 0x38;
   lcd_comand();

   P1 = 0x0e;
   lcd_comand();

   P1 = 0x06;
   lcd_comand();

   P1 = 0x01;
   lcd_comand();

   P1 = 0x02;
   lcd_comand();
         }
 /* function to send control word to LCD */
   void lcd_comand(void)
   {
```

```
   int i;
   rs = 0;
   e = 1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
/* function to send data word to LCD */
   void lcd_data(void)
   {
   int i;
   rs = 1;
   e = 1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
```

Experiment 23

AIM: 1. To interface 4½-digit A/D converter ICL7135 with microcontroller and
2. To measure analog voltage and display on LCD module.

COMPONENTS: 89C51, A/D converter ICL7135, precision voltage regulator LM329, 2-5.1V Zener diodes, 100k pot, 0.1µF, 0.47µF, 1.0µF metalized polyester capacitors (1 each), 1.0µF tantalum capacitor, 10µF capacitor, 100kΩ multi-turn pot, 4–100kΩ, 8.2kΩ, 100Ω, 27Ω resistors, 12.0 MHz crystal, switch, LCD module, bread board and connecting wires.

EQUIPMENT: Power supply +5V/3A, ±12V/100mA.

INTRODUCTION: The Intersil ICL7135 is a 4½-digit A/D converter. It is a precision A/D converter with multiplexed BCD output and digit drivers, combines dual slope conversion. The A/D converter has the following features.

- Accuracy guaranteed to ±1 count over entire ±20,000 counts
- Guaranteed zero reading for zero volt input.
- 1pA typical input current.
- Single reference voltage required.
- Over range and under range signal available for auto range capability.
- Six auxiliary inputs/outputs are available for interfacing UARTS.
- Multiplexed BCD output.
- All outputs are TTL compatible.
- Blinking output gives visual indication of over range.

Hardware: Figure 25 shows interfacing of 4½-digit A/D converter ICL7135 with the microcontroller 89C51 through port-0 of microcontroller. A stable 1.0000V reference voltage source is derived from a precision voltage regulator LM329 (6.9V) and applied to the reference input of A/D converter. The +5V and –5V voltages for A/D converter are generated using 5.1V zener diodes. A metalized polyester capacitor C (0.47µF) acts as an integrator. The 100kΩ and 0.1µF capacitor at the input pins IN HI and IN LO are used to prevent any noise picked up on the input leads, from entering the

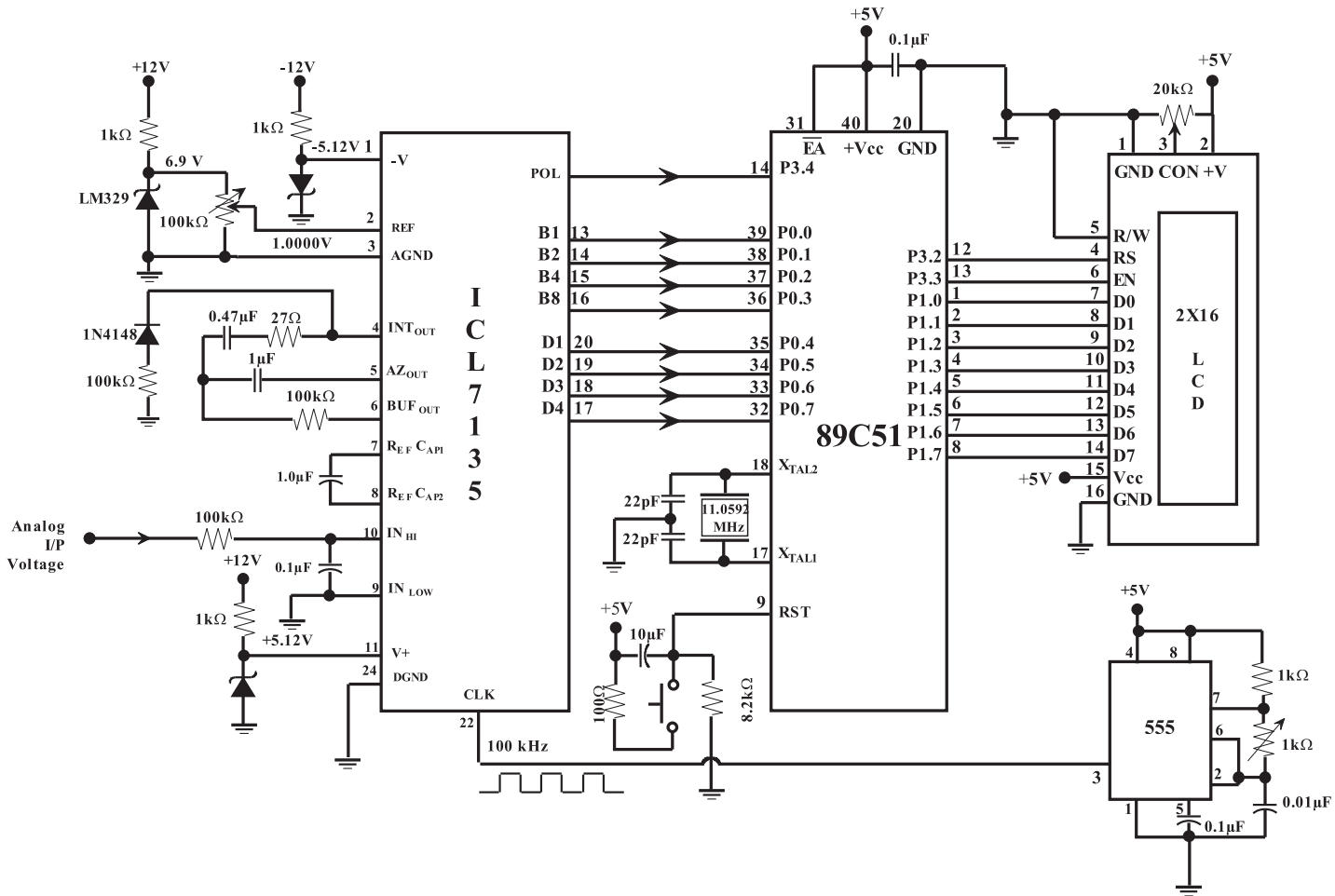


Fig. 25. Interfacing of 4½-digit A/D converter ICL7135 with the microcontroller.

IC. A clock of 100 kHz is generated for A/D converter using IC NE555 timer. The port-0 of microcontroller is used to read the digit strobes and BCD data from A/D converter. The output of the A/D converter is in multiplexed BCD form. The BCD data is available on B1, B2, B4, and B8 data lines of the converter. Logic HIGH on one of the digit strobe lines (D5, D4, D3, D2, and D1) indicates the presence of BCD code for corresponding digit on the data lines. The data from this A/D converter are read by polling the bit corresponding to a strobe line (until the bit goes high) and store the data in one of the registers for future reference. After reading over the BCD code for one digit, the bit corresponding to the strobe line for the next digit is polled until it goes logic HIGH and stored in another register. The process is repeated until the data for all the 5 digits are acquired. The circuit diagram also presents the interfacing of the LCD display module with the microcontroller. The data bus D0-D7 of display module is connected to P1.0-P1.7 of microcontroller and the control pins RS and E are connected to the P3.2 and P3.3 pins of 89C51 respectively.

Software: The software first initializes the stack pointer and LCD module. After initialization the program then calls the A/D conversion subroutine to read the analog voltage through A/D converter ICL7135. This digital data from A/D converter is displayed on the LCD module. The assembly language program and 'C' program developed in Raisonance Integrated Development Environment (RIDE) are given in program-27.

a. Initialize stack pointer
b. Initialize LCD module
c. Call subroutine to display 'Voltage=' on LCD module
d. Call subroutine to convert analog voltage into digital data (BCD)
e. Call subroutine to convert BCD to ASCII code and display voltage on LCD module
f. Call delay subroutine
g. Send control word to the LCD module to bring the cursor to the left corner of the module
h. Repeat the above steps from (c).

Procedure

1. Wire the circuit as shown in Fig. 25.
2. Load the assembly language program-27 or 'C' program into the program memory from the starting address 0000H.
3. Execute the program by powering to the circuit. Observe the display of the analog input voltage on LCD module.
4. Vary the analog input voltage from 0-2V in steps of 0.2V and record the corresponding digital data.
5. Tabulate the results in Table 14 and Table 15.

Table 14

S.No.	Analog voltage DVM (Volts)	Voltage Displayed on LCD module
1	0.0000	0.0000
2	0.1006	0.1008
3	0.3377	0.3380
4	0.6151	0.6154
5	0.8286	0.8289
6	1.0001	1.0002
7	1.2546	1.2547
8	1.5678	1.5679
9	1.7523	1.7524
10	2.0000	1.9999

Table 15

S.No.	Analog voltage DVM (Volts)	Voltage Displayed on LCD module
1	0.0000	0.0000
2	-0.1005	-0.1003
3	-0.2755	-0.2751
4	-0.5376	-0.5373
5	-0.7585	-0.7584
6	-1.1150	-1.1148
7	-1.3545	-1.3540
8	-1.6010	-1.6005
9	-1.8899	-1.8901
10	-2.0001	-1.9998

Program 27 : Program for converting analog voltage into digital using 4½ digit A/D converter ICL7135

Address	Opcode	Label	Mnemonic	Operand	Comments
		org	0000h		;starting address of the program
					;Initialization of LCD display
0000	75 81 40	mov		sp,#40h	;Initialize stack pointer
					;with starting address 40h
0003	74 38	mov		a,#38h	;function set control word
0005	12 00 6A	lcall		enable	;send to LCD module
0008	74 0E	mov		a,#0eh	;display ON/OFF control word
000A	12 00 6A	lcall		enable	;send to LCD module
000D	74 06	mov		a,#06h	;entry mode set control word
000F	12 00 6A	lcall		enable	;send to LCD module
0012	74 01	mov		a,#01h	;clear display control word
0014	12 00 6A	lcall		enable	;send to LCD module and
0017	74 02	mov		a,#02h	;return home control word
0019	12 00 6A	lcall		enable	;send to LCD module l1:
001C	12 00 94	lcall		v_display	;call subroutine to display
					;'voltage=' on LCD module

Program 27 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
001F	12 00 2D	lcall		adc	;call subroutine A/D conversion
0022	12 00 BD	lcall		display	;call subroutine to display ;voltage in terms of V
0025	74 02	mov		a,#02h	;bring cursor to left
0027	12 00 6A	lcall		enable	;corner of the display
002A	02 00 1C	ljmp		l1	;repeat the program ;program for ICL7135 A/D converter adc:
002D	E5 80	x1: mov		a,p0	;read digit strobes and ;data from A/D converter
002F	54 F0	anl		a,#0f0h	;mask data
0031	B4 00 F9	cjne		a,#00h,x1	;check whether D1-D4 digit ;strobes are LOW, if not ;go to location x1
0034	E5 80	mov		a,p0	;if all digit strobes are ;LOW, then read data from ;A/D converter
0036	54 0F	anl		a,#0fh	;mask digit strobes
0038	F9	mov		r1,a	;store MSD data in r1
0039	E5 80	x2: mov		a,p0	;read digit strobes and ;data from A/D converter
003B	54 F0	anl		a,#0f0h	;mask data
003D	B4 80 F9	cjne		a,#80h,x2	;poll digit strobe D4 until ;it goes HIGH
0040	E5 80	mov		a,p0	;read digital data
0042	54 0F	anl		a,#0fh	;mask digit strobes
0044	FA	mov		r2,a	;store data in r2
0045	E5 80	x3: mov		a,p0	;read digit strobes and ;data from A/D converter
0047	54 F0	anl		a,#0f0h	;mask data
0049	B4 40 F9	cjne		a,#40h,x3	;poll digit strobe D3 until ;it goes HIGH
004C	E5 80	mov		a,p0	;read digital data
004E	54 0F	anl		a,#0fh	;mask digit strobes
0050	FB	mov		r3,a	;store data in r3
0051	E5 80	x4: mov		a,p0	;read digit strobes and ;data from A/D converter
0053	54 F0	anl		a,#0f0h	;mask data
0055	B4 20 F9	cjne		a,#20h,x4	;poll digit strobe D2 until ;it goes HIGH
0058	E5 80	mov		a,p0	;read digital data
005A	54 0F	anl		a,#0fh	;mask digit strobes
005C	FC	mov		r4,a	;store data in r4
005D	E5 80	x5: mov		a,p0	;read digit strobes and ;data from A/D converter
005F	54 F0	anl		a,#0f0h	;mask data

Program 27 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0061	B4 10 F9	cjne		a,#10h,x5	;poll digit strobe D1 until ;it goes HIGH
0064	E5 80	mov		a,p0	;read digital data
0066	54 0F	anl		a,#0fh	;mask digit strobes
0068	FD	mov		r5,a	;store data in r5
0069	22	ret			;return to main program enable:
006A	F5 90	mov		p1,a	;send control word to display
006C	C2 B2	clr		p3.2	;RS=0
006E	D2 B3	setb		p3.3	;enable=1
0070	75 F0 FF	mov		b,#0ffh	;small delay
0073	D5 F0 FD	djnz		b,$	
0076	C2 B3	clr		p3.3	;enable=0
0078	75 F0 FF	mov		b,#0ffh	;small delay
007B	D5 F0 FD	djnz		b,$	
007E	22	ret			;return to main program data_disp:
007F	F5 90	mov		p1,a	;send character code to display
0081	D2 B2	setb		p3.2	;RS=1
0083	D2 B3	setb		p3.3	;enable=1
0085	75 F0 FF	mov		b,#0ffh	;small delay
0088	D5 F0 FD	djnz		b,$	
008B	C2 B3	clr		p3.3	;enable=0
008D	75 F0 FF	mov		b,#0ffh	;small delay
0090	D5 F0 FD	djnz		b,$	
0093	22	ret			;return to main program v_display:
0094	74 56	mov		a,#56h	;code for 'V'
0096	12 00 7F	lcall		data_disp	;display it on LCD module
0099	74 6F	mov		a,#6fh	;code for 'o'
009B	12 00 7F	lcall		data_disp	;display it on LCD module
009E	74 6C	mov		a,#6ch	;code for 'l'
00A0	12 00 7F	lcall		data_disp	;display it on LCD module
00A3	74 74	mov		a,#74h	;code for 't'
00A5	12 00 7F	lcall		data_disp	;display it on LCD module
00A8	74 61	mov		a,#61h	;code for 'a'
00AA	12 00 7F	lcall		data_disp	;display it on LCD module
00AD	74 67	mov		a,#67h	;code for 'g'
00AF	12 00 7F	lcall		data_disp	;display it on LCD module
00B2	74 65	mov		a,#65h	;code for 'e'
00B4	12 00 7F	lcall		data_disp	;display it on LCD module
00B7	74 3D	mov		a,#3dh	;code for '='
00B9	12 00 7F	lcall		data_disp	;display it on LCD module
00BC	22	ret			;return to main program display:
00BD	20 B4 08	jb		p3.4,neg	
00C0	74 2B	mov		a,#2bh	;+ve sign
00C2	12 00 7F	lcall		data_disp	
00C5	02 00 CD	ljmp		skp	
00C8	74 2D	neg:	mov	a,#2dh	;-ve sign
00CA	12 00 7F	lcall		data_disp	skp:
00CD	E8	mov		a,r0	;read most significant digit
00CE	44 30	orl		a,#30h	;convert it into ASCII
00D0	12 00 7F	lcall		data_disp	;display it on LCD

Program 27 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
00D3	74 2E	mov		a,#2eh	;load acc. with code for '.'
00D5	12 00 7F	lcall		data_disp	;display it on LCD
00D8	E9	mov		a,r1	;read next digit
00D9	44 30	orl		a,#30h	;convert it into ASCII
00DB	12 00 7F	lcall		data_disp	;display it on LCD
00DE	EA	mov		a,r2	;read next digit
00DF	44 30	orl		a,#30h	;convert it into ASCII
00E1	12 00 7F	lcall		data_disp	;display it on LCD
00E4	EB	mov		a,r3	;read next digit
00E5	44 30	orl		a,#30h	;convert it into ASCII
00E7	12 00 7F	lcall		data_disp	;display it on LCD
00EA	EC	mov		a,r4	;read next digit
00EB	44 30	orl		a,#30h	;convert it into ASCII
00ED	12 00 7F	lcall		data_disp	;display LSD on LCD
00F0	22	ret			;return to main program
					;delay subroutine delay:
00F1	7D 05	mov		r5,#05h	
00F3	7F FF	l8: mov		r7,#0ffh	
00F5	7E FF	l7: mov		r6,#0ffh	
00F7	DE FE	djnz		r6,$	
00F9	DF FA	djnz		r7,l7	
00FB	DD F6	djnz		r5,l8	
00FD	22	ret			
		end			;program ends

```
/* program for ICL7135 A/D converter */

#include <stdio.h>
#include <reg51.h>

   sbit   x2  =  P0^7;
   sbit   x3  =  P0^6;
   sbit   x4  =  P0^5;
   sbit   x5  =  P0^4;

   sbit  rs  =  P3^2;
   sbit   e  =  P3^3;
   sbit  pol=  P3^4;

         int  d1,d2,d3,d4,d5;
```

```
   void main (void)
   {
   void adc(void);
   void lcd_comand();
   void lcd_data();
   void lcd_int(void);
   int i;
   int v[] = {'V','O','L','T','A','G','E','='};
   lcd_int();     //initialization of LCD module

rep1:
         P1 = 0x02;    //bring cursor to home
         lcd_comand();

   for (i = 0;i<8;i++)
   {
   P1 = v[i];    //display label 'VOLTAGE=' ON LCD module
   lcd_data();
   }
   adc();         //A/D conversion subroutine
       display();   //display voltage
   goto rep1;    //repeat above steps continuously
   }
   void adc(void)
   {
   do{
   d1 = P0;
   d1 = d1 & 0xf0;
   }while (d1!=0);
   d1 = P0 & 0x01;

   while (x2!=1);
   d2 = P0 & 0x0f;

   while (x3!=1);
   d3 = P0 & 0x0f;

   while (x4!=1);
   d4 = P0 & 0x0f;

   while (x5!=1);
   d5 = P0 & 0x0f;
   }
```

```
void display(void)
{
if(pol == 1)
{
P1 = '-'
Lcd_data();
}
d1 = d1 | 0x30;
P1 = d1;
lcd_data();

P1 = 0x2e;
lcd_data();
d2 = d2 | 0x30;

P1 = d2;
lcd_data();
d3 = d3 | 0x30;

P1 = d3;
lcd_data();
d4 = d4 | 0x30;

P1 = d4;
lcd_data();
d5 = d5 | 0x30;

P1 = d5;
lcd_data();
}

/*LCD initialization subroutine */
void lcd_int(void)
{
P1 = 0x38;
lcd_comand();

P1 = 0x0e;
lcd_comand();
P1 = 0x06;
lcd_comand();

P1 = 0x01;
lcd_comand();
```

```
   P1 = 0x02;
   lcd_comand();
   }

/* subroutine for sending control words to LCD module */
void lcd_comand(void)
   {
   int i;
   rs = 0;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }

/* subroutine for sending data to display on LCD module */
   void lcd_data(void)
   {
   int i;
   rs = 1;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
```

Experiment 24

AIM: To construct a temperature measurement system.

COMPONENTS: Transducer LM335, LM336, 4-Op-amp MC3140, A/D converter ICL7135, precision voltage regulator LM329, NE555 timer, 2-5.1V zener diodes, 0.1 μF, 0.47μF, 1.0μF (metalized polyester capacitors) each one, 1 kΩ, 100 kΩ multi-turn pot, 6-10 kΩ, 4–100 kΩ, 1–27Ω resistor, 8.2kΩ, 100Ω, resistors, switch, LCD module, bread board and connecting wires.

EQUIPMENT: Power supply +5V/3A, temperature controlled water bath.

INTRODUCTION: The temperature measurement plays an important role both in R & D and industry. The conventional temperature transducers have some significant limitations like non-linearity, low output etc., the IC sensor LM335 overcomes these difficulties. It operates on the principle of change in electrical output voltage as a function of temperature.

Hardware: Figure 26 shows the complete schematic diagram of microcontroller based temperature measurement system. The LM335 is a Kelvin temperature sensor i.e., its output is 2.73V±10mV/°K. In order to get the temperature in Degree Celsius, the voltage 2.73V is subtracted from the output of the sensor. Operational amplifier A1 produces a reference voltage of 2.73V, which is subtracted from the output of the sensor using an operational amplifier A2 in differential amplifier configuration as shown in Fig. 26. The differential amplifier produces a charge ±10mV/°C. The output of differential amplifier is given to the 4½-digit A/D converter. The schematic also shows the interfacing of ICL7135 A/D converter and LCD module with the microcontroller. The port-0 and port-1 of microcontroller are used to interface A/D converter and LCD modules respectively. A temperature controlled water bath is to be employed for performing the experiment.

Software: The software first initializes the stack pointer and LCD module. After initialization the program then calls the A/D conversion subroutine to read the voltage

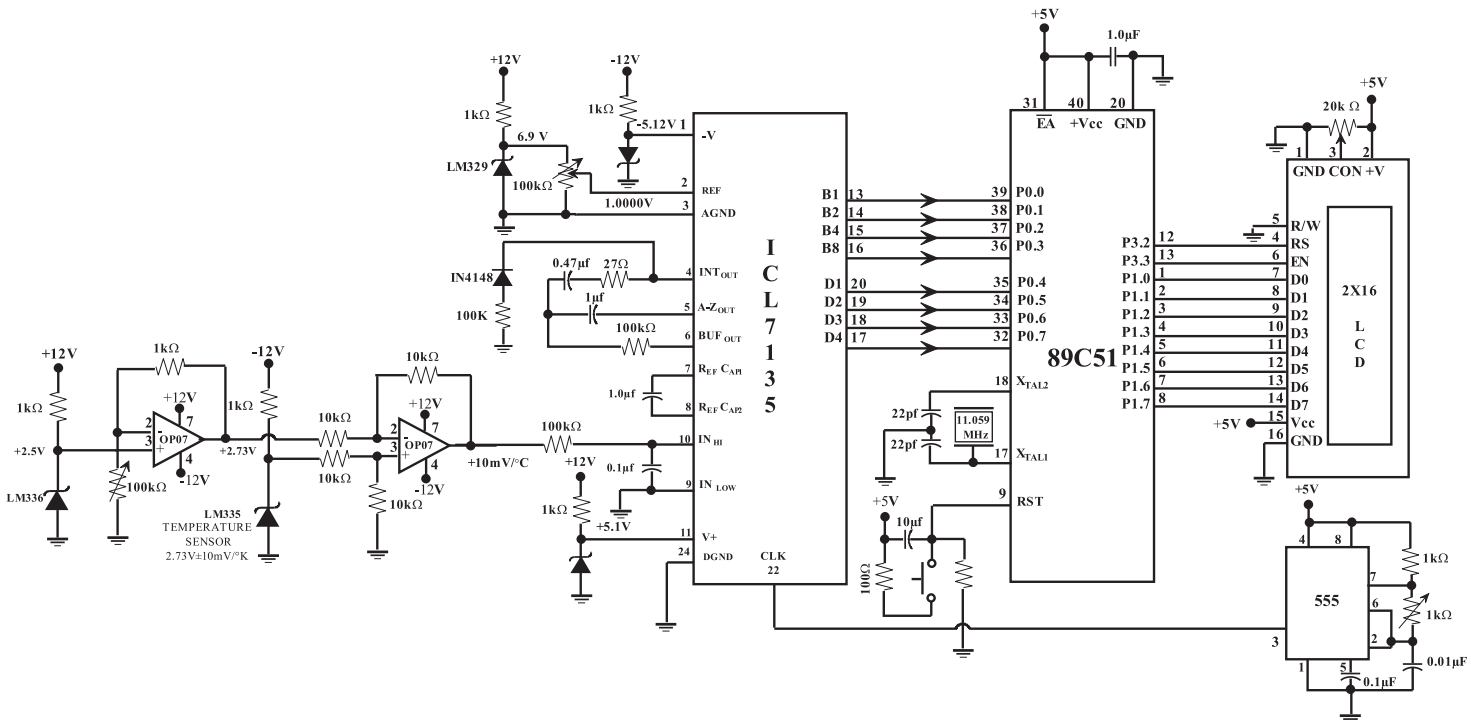


Fig. 26. The complete schematic diagram of microcontroller based temperature measurement system.

of the temperature sensor LM335 through analog to digital converter ICL7135. This digital data from A/D converter directly indicates the temperature of the body in Degree Celsius, same will be sent to the LCD module. The assembly program is given in program-28. The execution of the program-28 displays the temperature of the water bath.

a. Initialize the stack pointer
b. Initialize the LCD module
c. Call subroutine to display 'Temp.=' on LCD module
d. Call subroutine to convert analog voltage into digital data
e. Call subroutine to display temperature on LCD module
f. Call delay subroutine
g. Send control word to the LCD module to bring the cursor to the left corner of the module
h. Repeat the above steps from (c).

Procedure

1. Wire the circuit as shown in Fig. 26.
2. Load the program-28 into the program memory of microcontroller using EPROM programmer.
3. Execute the program by applying +5V to the circuit. Observe the continuous display of the Temperature = XX..XX on the LCD module.
4. Vary the temperature of water bath from 25 to 65°C in steps of 10°C and record the temperature displayed both on temperature bath and microcontroller based temperature measurement system output.
5. Tabulate the results in Table 16.

Table 16

S.No.	Water bath temperature in°C	Temperature displayed on LCD module in°C
1	25	25.5
2	35	35.0
3	45	45.4
4	55	55.2
5	65	65.0

Program 28 : Program for temperature measurement using LM335 sensor

Address	Opcode	Label	Mnemonic	Operand	Comments
		org	0000h		
					;Initialization of LCD display
0000	75 81 40	mov		sp,#40h	;Initialize stack pointer ;with starting address 40h
0003	74 38	mov		a,#38h	;function set control word
0005	12 00 64	lcall		enable	;send to LCD module
0008	74 0E	mov		a,#0eh	;display ON/OFF control word
000A	12 00 64	lcall		enable	;send to LCD module
000D	74 06	mov		a,#06h	;entry mode set control word
000F	12 00 64	lcall		enable	;send to LCD module
0012	74 01	mov		a,#01h	;clear display control word
0014	12 00 64	lcall		enable	;send to LCD module
0017	74 02	mov		a,#02h	;return home control word
0019	12 00 64	lcall		enable	;send to LCD module rep:
001C	12 00 8E	lcall		t_display	;call subroutine to display ;'Temp.=' on LCD module
001F	12 00 37	lcall		adc	;call subroutine A/D conversion
0022	12 00 AD	lcall		display	;call subroutine to display ;temperature
0025	74 DF	mov		a,#0dfh	;code for 'o'
0027	12 00 79	lcall		data_disp	;display it on LCD module
002A	74 43	mov		a,#43h	;code for 'C'
002C	12 00 79	lcall		data_disp	;display it on LCD module
002F	74 02	mov		a,#02h	;bring cursor to left
0031	12 00 64	lcall		enable	;corner of the display
0034	02 00 1C	ljmp		rep	;repeat the program ;program for ICL7135 A/D converter
				adc:	
0037	E5 80	x0: mov		a,p0	;read digit strobes and ;data from A/D converter
0039	54 F0	anl		a,#0f0h	;mask data
003B	B4 00 F9	cjne		a,#00h,x0	;check whether D1-D4 digit ;strobes are LOW, if not ;go to location x0
003E	E5 80	mov		a,p0	;if all digit strobes are ;LOW, the read data from A/D converter
0040	54 0F	anl		a,#0fh	;mask digit strobes
0042	F8	mov		r0,a	;store MSD data in r0

Program 28 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0043	30 84 FD	jnb		p0.4,$	;poll digit strobe D4 until ;it goes HIGH
0046	E5 80	mov		a,p0	;read digital data
0048	54 F0	anl		a,#0f0h	;mask digit strobes
004A	F9	mov		r1,a	;store data in r1
004B	30 85 FD	jnb		p0.5,$	;poll digit strobe D3 until ;it goes HIGH
004E	E5 80	mov		a,p0	;read digital data
0050	54 0F	anl		a,#0fh	;mask digit strobes
0052	FA	mov		r2,a	;store data in r2
0053	30 86 FD	jnb		p0.6,$	;poll digit strobe D2 until ;it goes HIGH
0056	E5 80	mov		a,p0	;read digital data
0058	54 0F	anl		a,#0fh	;mask digit strobes
005A	FB	mov		r3,a	;store data in r3
005B	30 87 FD	jnb		p0.7,$	;poll digit strobe D1 until ;it goes HIGH
005E	E5 80	mov		a,p0	;read digital data
0060	54 0F	anl		a,#0fh	;mask digit strobes
0062	FC	mov		r4,a	;store data in r4
0063	22	ret			;return to main program
				enable:	
0064	F5 90	mov		p1,a	;send control word to display
0066	C2 B4	clr		p3.2	;RS=0
0068	D2 B5	setb		p3.3	;enable=1
006A	75 F0 FF	mov		b,#0ffh	;small delay
006D	D5 F0 FD	djnz		b,$	
0070	C2 B5	clr		p3.3	;enable=0
0072	75 F0 FF	mov		b,#0ffh	;small delay
0075	D5 F0 FD	djnz		b,$	
0078	22			ret	;return to main program
				data_disp:	
0079	F5 90	mov		p1,a	;send character code to display
007B	D2 B4	setb		p3.4	;RS=1
007D	D2 B5	setb		p3.5	;enable=1
007F	75 F0 FF	mov		b,#0ffh	;small delay
0082	D5 F0 FD	djnz		b,$	
0085	C2 B5	clr		p3.5	;enable=0
0087	75 F0 FF	mov		b,#0ffh	;small delay
008A	D5 F0 FD	djnz		b,$	
008D	22			ret	;return to main program
				t_display:	
008E	74 74	mov		a,#74h	;code for ‘T’
0090	12 00 79	lcall		data_disp	;display it on LCD module
0093	74 65	mov		a,#65h	;code for ‘e’
0095	12 00 79	lcall		data_disp	;display it on LCD module
0098	74 6D	mov		a,#6dh	;code for ‘m’

Program 28 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
009A	12 00 79	lcall		data_disp	;display it on LCD module
009D	74 70	mov		a,#70h	;code for 'p'
009F	12 00 79	lcall		data_disp	;display it on LCD module
00A2	74 2E	mov		a,#2eh	;code for '.'
00A4	12 00 79	lcall		data_disp	;display it on LCD module
00A7	74 3D	mov		a,#3dh	;code for '='
00A9	12 00 79	lcall		data_disp	;display it on LCD module
00AC	22			ret	;return to main program
00AD				display:	
00AD	E9	mov		a,r1	;read most significant digit ;of temperature from r1
00AE	44 30	orl		a,#30h	;convert it into ASCII
00B0	12 00 79	lcall		data_disp	;display it on LCD
00B3					
00B3	EA	mov		a,r2	;read next digit
00B4	44 30	orl		a,#30h	;convert it into ASCII
00B6	12 00 79	lcall		data_disp	;display it on LCD
00B9	74 2E	mov		a,#2eh	;load acc. with code for '.'
00BB	12 00 79	lcall		data_disp	;display it on LCD
00BE	EB	mov		a,r3	;read next digit
00BF	44 30	orl		a,#30h	;convert it into ASCII
00C1	12 00 79	lcall		data_disp	;display it on LCD
00C4	EC	mov		a,r4	;read next digit
00C5	44 30	orl		a,#30h	;convert it into ASCII
00C7	12 00 79	lcall		data_disp	;display it on LCD
00CA	22			ret	;return to main program ;delay subroutine
				delay:	
00CB	7D 05	mov		r5,#05h	
00CD	7F FF	l8: mov		r7,#0ffh	
00CF	7E FF	l7: mov		r6,#0ffh	
00D1	DE FE	djnz		r6,$	
00D3	DF FA	djnz		r7,l7	
00D5	DD F6	djnz		r5,l8	
00D7	22			ret	
00D8		end			;end of the program

```
/* program for temperature measurement using LM335 sensor and
ICL7135 A/D converter */

#include <stdio.h>
#include <reg51.h>

   sbit   x2 = P0^7;
```

```
   sbit   x3 = P0^6;
   sbit   x4 = P0^5;
   sbit   x5 = P0^4;

   sbit rs = P3^2;
   sbit  e = P3^3;

        int d1,d2,d3,d4,d5;

   void main (void)
   {
   void adc_7135(void);
   void lcd_comand();
   void lcd_data();
   void lcd_int(void);
   int i;

   int t[] = {'T','e','m','p','.','='};
       int k[] = {'o','C'};
   lcd_int();
rep1:
     P1 = 0x02;
     lcd_comand();
/* to display 'Temp.=' on LCD module */
     for (i = 0;i<6;i++)
    {
     P1 = t[i];
     lcd_data();
    }
   adc_7135();
       display();

/* to display '°C' on LCD module */
   for (i = 0;i<2;i++)
    {
    P1 = k[i];
    lcd_data();
    }
   goto rep1;
   }
/*Sub-routine for A/D conversion */
 void adc_7135(void)
   {
   do{
```

```
   d1 = P0;
   d1 = d1 & 0xf0;
   }while (d1!=0);
   d1 = P0 & 0x01;

while (x2!=1);
   d2 = P0 & 0x0f;

   while (x3!=1);
   d3 = P0 & 0x0f;

   while (x4!=1);
   d4 = P0 & 0x0f;

   while (x5!=1);
   d5 = P0 & 0x0f;
   }
/* sub-routine to convert BCD to ASCII codes */
   Void display(void)
   {
   d2 = d2 | 0x30;

   P1 = d2;
   lcd_data();

   d3 = d3 | 0x30;

   P1 = d3;
   lcd_data();

   P1 = 0x2e;
   lcd_data();

   d4 = d4 | 0x30;

   P1 = d4;
   lcd_data();

   d5 = d5 | 0x30;

   P1 = d5;
   lcd_data();
   }
```

```
   /* Sub-routine to initialization of LCD module */
   void lcd_int(void)
   {
   P1 = 0x38;
   lcd_comand();

   P1 = 0x0e;
   lcd_comand();

   P1 = 0x06;
   lcd_comand();

   P1 = 0x01;
   lcd_comand();

   P1 = 0x02;
   lcd_comand();
   }

/* sub-routine to send control words to the LCD module */
void lcd_comand(void)
   {
   int i;
   rs = 0;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
/* sub-routine to send data to display on the LCD module */
   void lcd_data(void)
   {
   int i;
   rs = 1;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
```

Experiment 25

AIM: To measure the frequency of a TTL signal.

COMPONENTS: Microcontroller 89C51, 12.0 MHz quartz crystal, 8.2kΩ,100Ω, resistors, switch, 20kΩ multi-turn pot, 0.1μF decoupling capacitor, LCD module, bread board and connecting wires.

EQUIPMENT: Power supply + 5V/3A.

INTRODUCTION: The frequency of signal is defined as number of cycles per second. The principle involved in the frequency measurement is shown in Fig. 27. It operates on the principle of gating the input signal into the counter for a predetermined time. If the signal (whose frequency is to be measured) is gated into the counter for an exactly 1 second, the counter contents would be the frequency of the input signal. An AND/NAND gate can be used for gating the input signal. As long as the gating pulse is logic 1, the output of the AND gate is same as the input signal. When the gating pulse returns to logic 0, the output of the AND gate is logic 0. Thus, exactly for period of 1 second the input signal pulses are allowed at the output of the AND gate. These pulses are counted using counter. The same principle is implemented using timer-0 and timer-1 of 89C51 microcontroller. Here timer-1 is used to generate time base and timer-0 is used as a counter.

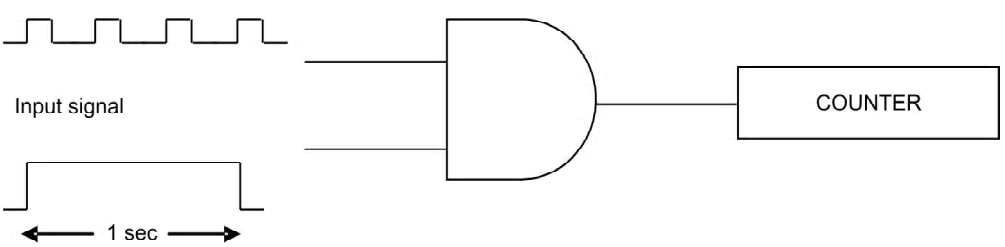


Fig. 27. Principle of frequency measurement.

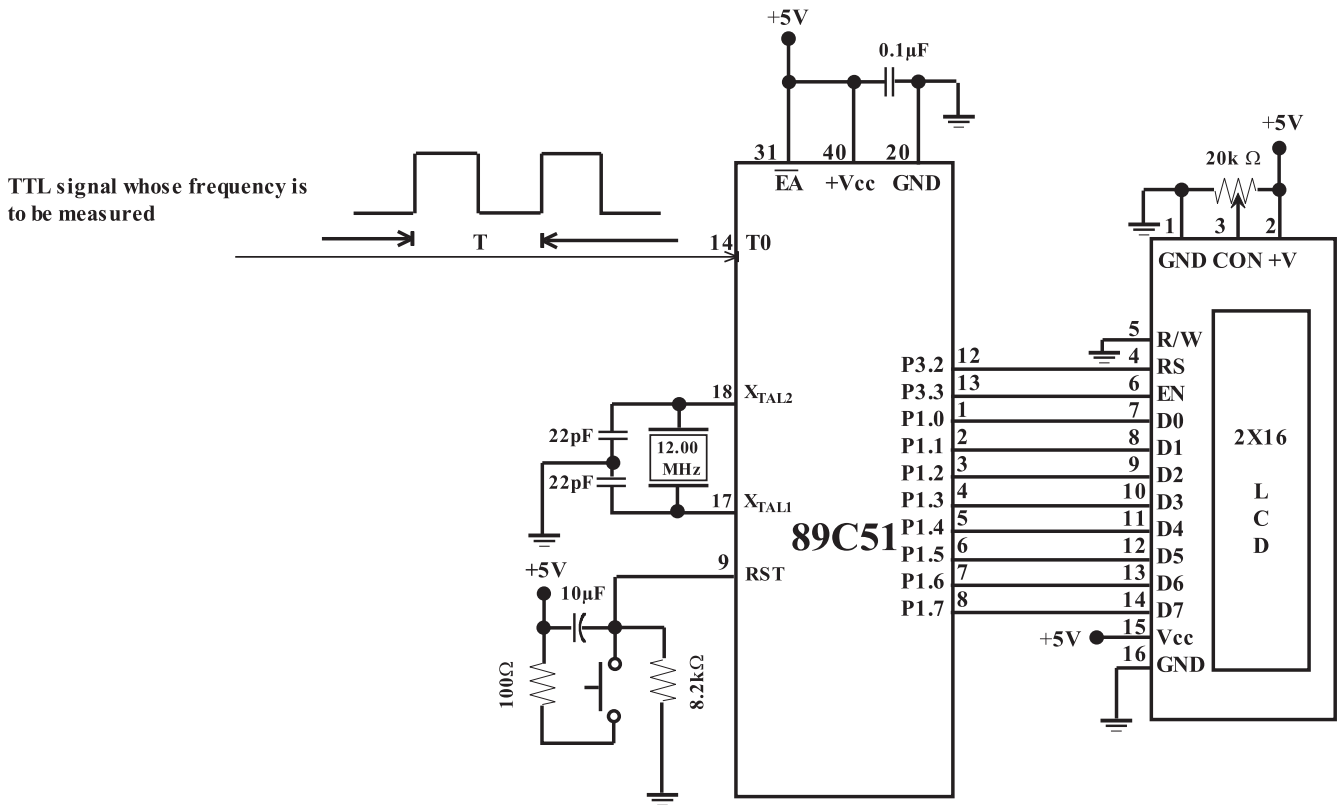


Fig. 28. The schematic diagram of measurement of frequency of TTL signal using microcontroller.

Hardware: The schematic diagram of measurement of frequency of TTL signal using microcontroller is shown in Fig. 28. The principle involved in the measurement is counting the number of cycles of TTL signal per second. The timer-0 and timer-1 of microcontroller are used for the measurement of frequency of a TTL signal. The TTL signal whose frequency is to be measured is applied to the P3.5 (T0) of the microcontroller. Initially, timer-1 is loaded with number 3CB0H (15536) to generate accurate 50 msec time base, and timer-0 is used as a 16-bit counter to count number of cycles of TTL signal in 50 msec duration. The frequency of the TTL signal is then calculated by the following equation:

Frequency = (counter contents)*20 Hz

For example, if the counter-1 content is 500 over an interval of 50 msec, then the frequency is:

Frequency = 500*20
Frequency = 10000 Hz = 10 kHz

the contents of counter-1 are multiplied with 20 and resulting number is converted into BCD number and sent to the LCD module to display the frequency of the input signal in Hz. The circuit diagram shown in Fig. 28 also presents the interfacing of the LCD display module with the microcontroller. The data bus D0-D7 of the display module is connected to P1.0-P1.7 of microcontroller and the control pins RS and E are connected to the P3.2 and P3.3 pins of 89C51 respectively.

Software: The software first initializes stack pointer and LCD module. After initialization, the program then initializes timer-1 as 16-bit timer in mode-1and it is loaded with 3CB0H for generating accurate 50 msec time-base and program initializes timer-0 as 16-bit counter in mode-1 for counting clock cycles of external signal in 50 msec duration. Then program starts counter-1 and timer-0 and it enables interrupts of timer-0 (ET0 and EA bits of interrupt enable register will be set) and it enters into infinite loop. After 50 msec, timer-1 overflows and generates an interrupt signal which will interrupt CPU and program control will be transferred to the location 001BH. Then the program control again transferred to the location 'int_freq' there it stops timer-1 and counter-0 and disables interrupts. The program reads the contents of counter-1 and multiplies with 20. Then the count is converted into BCD and sent to the display module to display frequency of TTL signal in Hz. For continuous measurement program re-initializes timer-1 and counter-0 and enables interrupts. The program control then returns to the main program.

a. Initialize the stack pointer
b. Initialize the LCD module
c. Call subroutine to display 'Freq=' on LCD module
d. Initialize timer-0 as 16-bit counter and timer-1 as 16-bit timer in mode-1
e. Load timer-1 with 3CB0H to produce 50 msec delay (time base)

f. Load counter-0 with 0000H for counting external signal frequency
g. Start timer-1 and timer-0
h. Set timer-1 interrupts ET1 bit and EA bit
i. Enter into infinite loop
j. When timer-1 interrupt occurs, then stop counter-1 and disable interrupts
k. Read counter-0 contents and multiply with 2
l. Call subroutine to convert the number into BCD number
m. Send control word to the LCD module to bring the cursor to left corner of the module
n. Call subroutine to display on frequency of the signal on LCD module
o. Call delay subroutine
p. Initialize timer-1, timer-0 and re-enable timer-0 interrupt
q. Return to main program.

Table 18

S.No.	Frequency DVM (kHz)	Frequency Displayed on LCD module (kHz)
1	0.010	000.00
2	0.101	000.10
3	0.500	000.50
4	1.000	001.00
5	10.00	010.00
6	100.0	100.00

Program 29 : Program for measurement of frequency of TTL signal

Address	Opcode	Label	Mnemonic	Operand	Comments
		org	0000h		
0000	02 00 1E	ljmp		main	;jump to main program
		org	001bh		;timer-1 interrupt location
001B	02 00 53	ljmp		int_freq	
				main:	
					;Initialization of LCD display
001E	75 81 40	mov		sp,#40h	;Initialize stack pointer
					;with starting address 40h
0021	74 38	mov		a,#38h	;function set control word
0023	12 00 A3	lcall		enable	;send to LCD module
0026	74 0E	mov		a,#0eh	;display ON/OFF control word
0028	12 00 A3	lcall		enable	;send to LCD module
002B	74 06	mov		a,#06h	;entry mode set control word
002D	12 00 A3	lcall		enable	;send to LCD module
0030	74 01	mov		a,#01h	;clear display control word
0032	12 00 A3	lcall		enable	;send to LCD module
0035	74 02	mov		a,#02h	;return home control word

Program 29 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0037	12 00 A3	lcall		enable	;send to LCD module
003A	75 89 15	mov		tmod,#15h	;initialize timer-0 as 16-bit timer ;and timer-1 as 16-bit counter
003D	75 8B B0	mov		tl1,#0b0h	;load low byte of timer-1 with B0H
0040	75 8D 3C	mov		th1,#3ch	;load high byte of timer-1 with 3CH
0043	75 8C 00	mov		th0,#00h	;clear low byte of timer-0
0046	75 8A 00	mov		tl0,#00h	;clear high byte of timer-0
0049	D2 8C	setb		tr0	;start timer-0
004B	D2 8E	setb		tr1	;start counter-1
004D	D2 AB	setb		et1	;enable timer-1 interrupt
004F	D2 AF	setb		ea	;enable all interrupt flag
0051	80 FE	sjmp		$	;enter into infinite loop ;interrupt service routine int_freq:
0053	C2 8C	clr		tr0	;stop timer-0
0055	C2 8E	clr		tr1	;stop counter-1
0057	C2 AF	clr		ea	;disable all interrupts
0059	E5 8A	mov		a,tl0	;read low byte of timer-0
005B	F8	mov		r0,a	;store it in r0
005C	E5 8C	mov		a,th0	;read high byte of timer-1
005E	F9	mov		r1,a	;store it in r1
005F	7A 00	mov		r2,#00h	;clear r2
0061	C3	clr		c	;clear carry flag
0062	E8	mov		a,r0	;read r0
0063	33	rlc		a	;rotate accumulator right through ;carry flag
0064	F8	mov		r0,a	;store accumulator contents in r0
0065	E9	mov		a,r1	;read r1
0066	33	rlc		a	;rotate accumulator right through ;carry flag
0067	F9	mov		r1,a	;store acc. in r1
0068	EA	mov		a,r2	;read r2
0069	33	rlc		a	;rotate accumulator right through ;carry flag
006A	FA	mov		r2,a	;store acc. in r2
006B	12 01 18	lcall		hex_bcd	;call subroutine to convert HEX ;number BCD number
006E	74 02	mov		a,#02h	;bring cursor to left
0070	12 00 A3	lcall		enable	;corner of the display
0073	12 00 CD	lcall		f_display	;call subroutine to display 'Freq=' ;on LCD module
0076	12 00 E7	lcall		display	;call subroutine to display ;frequency in terms of KHz

Program 29 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0079	74 4B	mov		a,#4bh	;code for 'K'
007B	12 00 B8	lcall		data_disp	;display it on LCD module
007E	74 48	mov		a,#48h	;code for 'H'
0080	12 00 B8	lcall		data_disp	;display it on LCD module
0083	74 71	mov		a,#71h	;code for 'z'
0085	12 00 B8	lcall		data_disp	;display it on LCD module
0088	12 01 0B	lcall		delay	;call delay subroutine
008B	75 89 15	mov		tmod,#15h	;initialize timer-1 as 16-bit timer ;and timer-0 as 16-bit counter
008E	75 8B B0	mov		tl1,#0b0h	;load low byte of timer-1 with B0H
0091	75 8D 3C	mov		th1,#3ch	;load high byte of timer-1 with 3CH ;for 50 msec delay
0094	75 8C 00	mov		th0,#00h	;clear low byte of counter-0
0097	75 8A 00	mov		tl0,#00h	;clear high byte of counter-0 ;for counting pulses
009A	D2 8C	setb		tr0	;start timer-0
009C	D2 8E	setb		tr1	;start counter-1
009E	D2 AB	setb		et1	;enable timer-0 interrupt in IE
00A0	D2 AF	setb		ea	;enable all interrupt flag in IE
00A2	32			reti	;return to main program
				enable:	
00A3	F5 90	mov		p1,a	;send control word to display
00A5	C2 B2	clr		p3.2	;RS=0
00A7	D2 B3	setb		p3.3	;enable=1
00A9	75 F0 FF	mov		b,#0ffh	;small delay
00AC	D5 F0 FD	djnz		b,$	
00AF	C2 B3	clr		p3.3	;enable=0
00B1	75 F0 FF	mov		b,#0ffh	;small delay
00B4	D5 F0 FD	djnz		b,$	
00B7	22			ret	;return to main program
				data_disp:	
00B8	F5 90	mov		p1,a	;send character code to display
00BA	D2 B2	setb		p3.2	;RS=1
00BC	D2 B3	setb		p3.3	;enable=1
00BE	75 F0 FF	mov		b,#0ffh	;small delay
00C1	D5 F0 FD	djnz		b,$	
00C4	C2 B3	clr		p3.3	;enable=0
00C6	75 F0 FF	mov		b,#0ffh	;small delay
00C9	D5 F0 FD	djnz		b,$	
00CC	22	ret			;return to main program
				f_display:	
00CD	74 46	mov		a,#46h	;code for 'F'
00CF	12 00 B8	lcall		data_disp	;display it on LCD module
00D2	74 72	mov		a,#72h	;code for 'r'
00D4	12 00 B8	lcall		data_disp	;display it on LCD module

Program 29 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
00D7	74 65	mov		a,#65h	;code for 'e'
00D9	12 00 B8	lcall		data_disp	;display it on LCD module
00DC	74 71	mov		a,#71h	;code for 'q'
00DE	12 00 B8	lcall		data_disp	;display it on LCD module
00E1	74 3D	mov		a,#3dh	;code for '='
00E3	12 00 B8	lcall		data_disp	;display it on LCD module
00E6	22	ret			;return to main program
				display:	
00E7	EB	mov		a,r3	;read most significant digit
00E8	44 30	orl		a,#30h	;convert it into ASCII
00EA	12 00 B8	lcall		data_disp	;display it on LCD
00ED	EC	mov		a,r4	;read next digit
00EE	44 30	orl		a,#30h	;convert it into ASCII
00F0	12 00 B8	lcall		data_disp	;display it on LCD
00F3	ED	mov		a,r5	;read next digit
00F4	44 30	orl		a,#30h	;convert it into ASCII
00F6	12 00 B8	lcall		data_disp	;display it on LCD
00F9	74 2E	mov		a,#2eh	;load acc. with code for '.'
00FB	12 00 B8	lcall		data_disp	;display it on LCD
00FE	EE	mov		a,r6	;read next digit
00FF	44 30	orl		a,#30h	;convert it into ASCII
0101	12 00 B8	lcall		data_disp	;display it on LCD
0104	EF	mov		a,r7	;read next digit
0105	44 30	orl		a,#30h	;convert it into ASCII
0107	12 00 B8	lcall		data_disp	;display LSD on LCD
010A	22	ret			;return to main program
					;delay program
				delay:	
010B	7D 08	mov		r5,#08h	
010D	7F FF	l8: mov		r7,#0ffh	
010F	7E FF	l7: mov		r6,#0ffh	
0111	DE FE	djnz		r6,$	
0113	DF FA	djnz		r7,l7	
0115	DD F6	djnz		r5,l8	
0117	22	ret			
					;input 24 bit hex number is in r2_hb,r1,r0_lb
					;output bcd number is in r0_hb,r1,r2,r3,r4,r5,r6,r7_lb
				hex_bcd:	
0118	7F 19	mov		r7,#19h	
011A	12 01 79	lcall		conv1	
011D	EB	mov		a,r3	
011E	FC	mov		r4,a	
011F	7F 19	mov		r7,#19h	
0121	12 01 79	lcall		conv1	
0124	EB	mov		a,r3	
0125	FD	mov		r5,a	

Program 29 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0126	7F 19	mov		r7,#19h	
0128	12 01 79	lcall		conv1	
012B	EB	mov		a,r3	
012C	FE	mov		r6,a	
012D	7F 19	mov		r7,#19h	
012F	12 01 79	lcall		conv1	
0132	EB	mov		a,r3	
0133	FF	mov		r7,a	
0134	EC	mov		a,r4	
0135	54 0F	anl		a,#0fh	
0137	F8	mov		r0,a	
0138	EC	mov		a,r4	
0139	54 F0	anl		a,#0f0h	
013B	C4	swap		a	
013C	F9	mov		r1,a	
013D	ED	mov		a,r5	
013E	54 0F	anl		a,#0fh	
0140	FA	mov		r2,a	
0141	ED	mov		a,r5	
0142	54 F0	anl		a,#0f0h	
0144	C4	swap		a	
0145	FB	mov		r3,a	
0146	EE	mov		a,r6	
0147	54 0F	anl		a,#0fh	
0149	FC	mov		r4,a	
014A	EE	mov		a,r6	
014B	54 F0	anl		a,#0f0h	
014D	C4	swap		a	
014E	FD	mov		r5,a	
014F	EF	mov		a,r7	
0150	54 0F	anl		a,#0fh	
0152	FE	mov		r6,a	
0153	EF	mov		a,r7	
0154	54 F0	anl		a,#0f0h	
0156	C4	swap		a	
0157	FF	mov		r7,a	
0158	EF	mov		a,r7	
0159	F5 08	mov		08h,a	
015B	EE	mov		a,r6	
015C	F5 09	mov		09h,a	
015E	ED	mov		a,r5	
015F	F5 0A	mov		0ah,a	
0161	EC	mov		a,r4	
0162	F5 0B	mov		0bh,a	
0164	E8	mov		a,r0	
0165	FF	mov		r7,a	
0166	E9	mov		a,r1	
0167	FE	mov		r6,a	

Program 29 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0168	EA	mov		a,r2	
0169	FD	mov		r5,a	
016A	EB	mov		a,r3	
016B	FC	mov		r4,a	
016C	E5 08	mov		a,08h	
016E	F8	mov		r0,a	
016F	E5 09	mov		a,09h	
0171	F9	mov		r1,a	
0172	E5 0A	mov		a,0ah	
0174	FA	mov		r2,a	
0175	E5 0B	mov		a,0bh	
0177	FB	mov		r3,a	
0178	22	ret			
0179	C3	conv1:	clr	c	
017A	74 00	mov		a,#00h	
017C	FB	mov		r3,a	
017D	C3	conv2:	clr	c	
017E	74 00	mov		a,#00h	
0180	DF 01	djnz		r7, cont	
0182	22	ret			
				cont:	
0183	E8	mov		a,r0	
0184	33	rlc		a	
0185	F8	mov		r0,a	
0186	E9	mov		a,r1	
0187	33	rlc		a	
0188	F9	mov		r1,a	
0189	EA	mov		a,r2	
018A	33	rlc		a	
018B	FA	mov		r2,a	
018C	EB	mov		a,r3	
018D	F5 F0	mov		b,a	
018F	35 F0	addc		a,b	
0191	D4	da		a	
0192	FB	mov		r3,a	
0193	50 E8	jnc		conv2	
0195	E8	mov		a,r0	
0196	34 00	addc		a,#00h	
0198	F8	mov		r0,a	
0199	E9	mov		a,r1	
019A	34 00	addc		a,#00h	
019C	F9	mov		r1,a	
019D	EA	mov		a,r2	
019E	34 00	addc		a,#00h	
01A0	FA	mov		r2,a	
01A1	02 01 7D	ljmp		conv2	
		end			

```
/* program for measurement of frequency of TTL signal using timer0
and timer1*/

#include <stdio.h>
#include <reg51.h>

   sbit   rs = P3^2;
   sbit    e = P3^3;
   int d,d1,d2,d3,d4,d5,d6,f;
   void main (void)
   {
       void lcd_int(void);
   void lcd_comand(void);
   void lcd_data(void);
   void data_conv(void);
   void display(void);
   int i, hc,lc;

   int c1[] = {'F','r','e','q','='};
   int c2[] = {'k','H','z'};
   lcd_int();          //initialization LCD module
   F0 = 0;             //clear user flag
   TMOD = 0x15;        //initialize timer-1 as 16-bit
                       //timer and timer-0 as 16-bit
                       //counter
   TH1 = 0xb0;         //load timer-1 with B03CH
   TL1 = 0x3c;         //for 50 msec delay

   TH0 = 0x00;         //clear counter-0 for counting
   TL0 = 0x00;         //input pulses of the signal
   TR0 = 1;            //start counter-0
   TR1 = 1;            //start timer-1
   ET1 = 1;            //enable timer-1 interrupt
   EA  = 1;            //enable master interrupt control
//bit

 /* program to read counter contents, convert it into frequency
     and display frequency on LCD module */
   while(1)
   {
   while(F0 == 0);
   F0 = 0;
   lc = TL0
   hc = TH0;
```

```
  f = hc*256+lc;
  data_conv();

     P1 = 0x02;
     lcd_comand();
     for (i = 1;i<6;i++) //display 'Freq=' on LCD module
      {
     P1 = c1[i];
     lcd_data();
      }
     display();
     for (i = 1;i<4;i++) //display 'kHz' on LCD module
      {
     P1 = c2[i];
     lcd_data();
      }
     display();

/* reinitialize the timers and enable interrupts */
    TMOD = 0x15;
    TH1 = 0xb0;
    TL1 = 0x3c;
    TH0 = 0x00;
    TL0 = 0x00;
    TR0 = 1;
    IT0 = 1;
    EX0 = 1;
    EA  = 1;
     }
     }
/* timer-1 interrupt sub-routine */
     void freq_isr(void) interrupt 3
  {
  EA  = 0;  //disable all interrupts
  TR0 = 0;  //stop timer-0
  TR1 = 0;  //stop timer-1
  F0  = 1;  //set user flag
  }
  void data_conv(void)
  {
   /* program to separate digits */

  d = f*20;       //to get frequency in Hz
  d1 = d/100000;
```

```
      d2 = d % 100000;    d2 = d2/10000;
   d3 = d % 10000;
   d3 = d3/1000;
   d4 = d % 1000;
   d4 = d4/100;
   d5 = d % 100;
   d5 = d5/10;
   d6 = d % 10;
   }
 /* Sub-routine to convert BCD to ASCII so as to display on LCD
*/
   void display(void)
   {
   d1 = d1+0x30;
   P1 = d1;
   lcd_data();

   P1 = 0x2e;
   lcd_data();

   d2 = d2+0x30;
   P1 = d2;
   lcd_data();

   d3 = d3+0x30;
   P1 = d3;
   lcd_data();

   d4 = d4+0x30;
   P1 = d4;
   lcd_data();

   d5 = d5+0x30;
   P1 = d5;
   lcd_data();

   d6 = d6+0x30;
   P1 = d6;
   lcd_data();
   }
     /* Sub-routine for initialization of LCD module */
   void lcd_int(void)
   {
```

```
P1 = 0x38;
lcd_comand();
P1 = 0x0e;
lcd_comand();
P1 = 0x06;
lcd_comand();

P1 = 0x01;
lcd_comand();
P1 = 0x02;
lcd_comand();
}

void lcd_comand(void)
{
int i;
rs = 0;
e =  1;
for (i = 1;i<1000;i++);
e = 0;
for (i = 1;i<1000;i++);
}
void lcd_data(void)
{
int i;
rs = 1;
e =  1;
for (i = 1;i<1000;i++);
e = 0;
for (i = 1;i<1000;i++);
}
```

Experiment 26

AIM: To measure pulse width (ON time) of a TTL signal

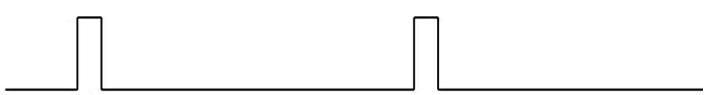
COMPONENTS: Microcontroller 89C51, monostable multivibrator 74LS121, 12.0MHz Quartz crystal, 8.2kΩ, 100Ω, resistors, switch, 20kΩ multi-turn pot, 0.1μF decoupling capacitors, LCD module, bread board and connecting wires.

EQUIPMENT: Variable capacitance box, power supply + 5V/3A.

INTRODUCTION: A mono-stable multi-vibrator is also called the one-shot multi-vibrator, this device has only one stable state. When triggered, it changes from its stable state to unstable state and remains there for a specified length of time before returning automatically to its stable state. The length of the output pulse is determined by a RC network. The ON time of the output of mono-stable multi-vibrator is given by the equation:

$$T = 0.7\ RC$$

The IC 74LS121 mono-stable multi-vibrator is used for generating different ON time TTL signal as shown below.



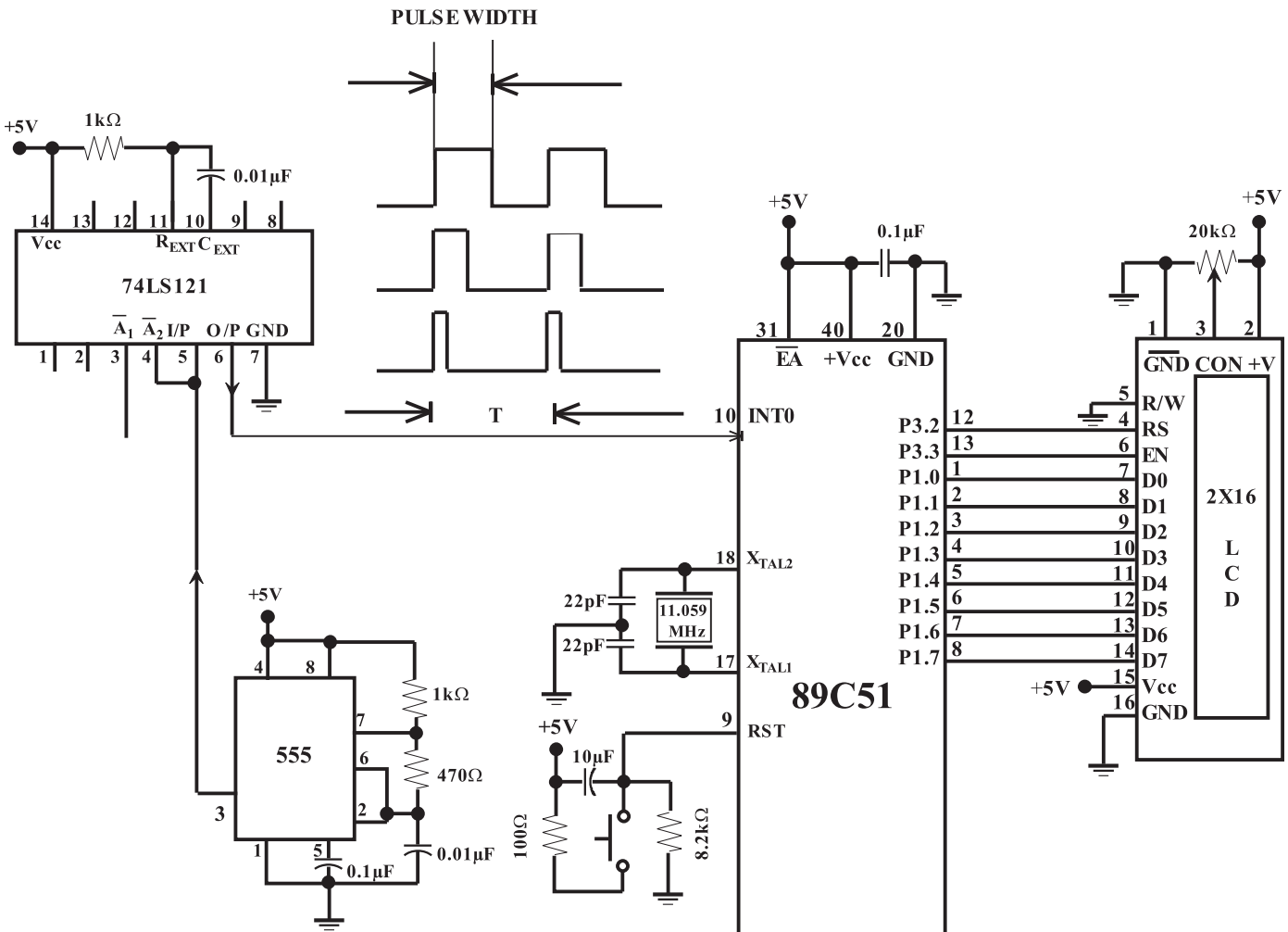


Fig. 29. The circuit diagram of microcontroller based pulse width (ON time) measurement system.

Hardware: The circuit diagram of microcontroller based pulse width (ON time) measurement system is shown in Fig. 29. A TTL signal whose ON time is to be measured is applied to the external interrupt line INT0 of microcontroller. The timer-0 of microcontroller is used to measure the ON time of the TTL signal. The timer-0 is programmed in mode-1 and GATE bit in TMOD register is set. This makes the internal 1MHz clock to be available to timer-0 for counting only when INT0 pin is HIGH and TR0 in TCON register is set. The interrupt is programmed to occur on a HIGH-to-LOW edge on INT0 pin. Initially timer-0 is cleared and TR0 bit is set, the timer-0 begins counting when the input pulse goes HIGH, and stops when the pulse goes from HIGH-to-LOW transition. This HIGH-to-LOW transition of the pulse also generates interrupt signal to the microcontroller. The program control will be transferred to the interrupt service routine which reads the contents of timer-0. The contents of timer-0 directly gives the width of the pulse in μsec. The count is converted into BCD number and displayed the same on LCD module which is interfaced to microcontroller.

The circuit diagram also presents the interfacing of the LCD display module with the microcontroller. The data bus D0-D7 of the display module is connected to P1.0-P1.7 of microcontroller and the control pins RS and E are connected to the P3.2 and P3.3 pins of 89C51 respectively.

Software: The software first initializes stack pointer and LCD module. After initialization of LCD module, the program then initializes timer-0 as 16-bit timer in mode-1and GATE bit in TMOD register is set. Then timer-0 it is loaded with 0000H, then software sets TR0 bit in TCON register, when the INT0 input goes HIGH then timer-0 starts counting internal clock. The program then enables timer-0 interrupt by setting ET0 and EA in interrupt enable register. The external pulse HIGH-to-LOW transition stops the timer-0 and generates an interrupt signal which will interrupt CPU and program control will be transferred to the location 0003H from there program control again transferred to the location 'int_pulse_width' there it disables interrupts and reads the contents of timer-0. Then the count is converted into BCD and sent to the display module to display ON time in terms of msec. For continuous measurement, the program re-initializes the timer-0 and enables interrupts and then returns to main program.

a. Initialize stack pointer with the address 40H
b. Initialize LCD module
c. Call subroutine to display 'Pulse_width=' on LCD module
d. Initialize timer-0 as 16-bit timer by sending Mode word (09H)
e. Clear timer-0 for measuring ON time of TTL signal in terms of μsec.
f. Set timer-0 interrupt ET0 & EA bits
g. Enter into infinite loop
h. When timer-0 interrupt occurs, then read contents of timer-0
i. Call subroutine to convert the HEX number into BCD number
j. Call subroutine to display ON time of the signal on LCD module

k. Send control word to the LCD module to bring the cursor to the left corner of the module
l. Call delay subroutine
m. Poll INT0 input until it goes HIGH
n. Poll INTO input until it goes LOW to start measurement exactly from LOW to HIGH transition
o. Initialize timer-0 and enable it's interrupt
p. Return to main program.

Procedure

1. Wire the circuit as shown in Fig. 29.
2. Load the program–30 (Assembly or 'C') into the program memory from the starting address 0000H using 89C51 programmer.
3. Execute the program by supplying + 5V power supply to the circuit.
4. Observe the display of ON time of the input signal on LCD module in msec.
5. Vary the ON time of the input signal by varying potentiometer connected to the mono-stable multi-vibrator and record the corresponding data displayed on LCD module.
6. Tabulate the results in Table 19:

Table 18

S.No.	ON time Theoretical (msec)	ON time Displayed on LCD module (msec)
1	0.045	0.046
2	0.088	0.088
3	0.128	0.129
4	1.000	1.001
5	10.00	10.00

Program 30 : Program for measurement of pulse width of a TTL signal

Address	Opcode	Label	Mnemonic	Operand	Comments
0000		org	0000h		
0000	02 00 06	ljmp		start	
0003	02 00 33	ljmp		int_pulse_width	
0006				start:	
					;Initialization of LCD display
0006	75 81 40	mov		sp,#40h	;Initialize stack pointer
					;with starting address 40h
0009	74 38	mov		a,#38h	;function set control word
000B	12 00 79	lcall		enable	;send to LCD module
000E	74 0E	mov		a,#0eh	;display ON/OFF control word
0010	12 00 79	lcall		enable	

Program 30 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0013	74 06	mov		a,#06h	;entry mode set control word
0015	12 00 79	lcall		enable	
0018	74 01	mov		a,#01h	;clear display control word
001A	12 00 79	lcall		enable	
001D	74 02	mov		a,#02h	;return home control word
001F	12 00 79	lcall		enable	
0022	75 89 09	mov		tmod,#09h	;timer-0 is programmed as 16- ;bit timer in mode-1 and GATE ;bit is set
0025	75 8C 00	mov		th0,#00h	;clear timer-0
0028	75 8A 00	mov		tl0,#00h	
002B	75 88 11	mov		tcon,#11h	;set TR0 & IT0 bits in TCON reg ;program INT0 as HIGH to LOW ;edge sensitive
002E	75 A8 81	mov		ie,#81h	;enable EX0 interrupt ;enable EA bit in IE register
0031	80 FE	sjmp		$	;stay in continuous loop
				int_pulse_width:	
0033	75 A8 00	mov		ie,#00	;disable all interrupts
0036	E5 8A	mov		a,tl0	;read low byte of timer-0
0038	F8	mov		r0,a	;store it in r0
0039	E5 8C	mov		a,th0	;read high byte of timer-0
003B	F9	mov		r1,a	;store it in r1
003C	7A 00	mov		r2,#00h	;clear r2
003E	75 88 00	mov		tcon,#00h	;clear timer-0 run bit
0041	12 00 E8	lcall		hex_bcd	;call subroutine to convert ;HEX number from timer-0 ;into BCD number
0044	74 02	mov		a,#02h	;bring cursor left corner of
0046	12 00 79	lcall		enable	;the display
0049	12 00 A3	lcall		t_display	;call subroutine to display 'T_on'
004C	12 00 BD	lcall		display	;on LCD module
004F	74 6D	mov		a,#6dh	;code for 'T'
0051	12 00 8E	lcall		data_disp	;display it on LCD module
0054	74 73	mov		a,#73h	;code for '_'
0056	12 00 8E	lcall		data_disp	;display it on LCD module
0059	74 65	mov		a,#65h	;code for 'o'
005B	12 00 8E	lcall		data_disp	;display it on LCD module
005E	74 63	mov		a,#63h	;code for 'n'
0060	12 00 8E	lcall		data_disp	;display it on LCD module
0063	30 B2 FD	jnb		int0,$	;poll INT0 pin continuously ;until it goes HIGH
0066	20 B2 FD	jb		int0,$	;poll INT0 pin continuously ;until it goes LOW

Program 30 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0069	75 89 09	mov		tmod,#09h	;timer-0 is programmed as 16- ;bit timer in mode-1 and GATE ;bit is set
006C	75 8C 00	mov		th0,#00h	;clear timer-0
006F	75 8A 00	mov		tl0,#00h	
0072	75 88 11	mov		tcon,#11h	;set TR0 bit in TCON register ;program INT0 as HIGH to LOW ;edge sensitive
0075	75 A8 81	mov		ie,#81h	;enable INT0 interrupt ;enable EA bit in IE register
0078	32	reti			;return to main program
0079				enable:	
0079	F5 90	mov		p1,a	;send control word to display
007B	C2 B4	clr		p3.2	;RS=0
007D	D2 B5	setb		p3.3	;enable=1
007F	75 F0 FF	mov		b,#0ffh	;small delay
0082	D5 F0 FD	djnz		b,$	
0085	C2 B5	clr		p3.3	;enable=0
0087	75 F0 FF	mov		b,#0ffh	;small delay
008A	D5 F0 FD	djnz		b,$	
008D	22	ret			;return to main program
008E				data_disp:	
008E	F5 90	mov		p1,a	;send character code to display
0090	D2 B4	setb		p3.2	;RS=1
0092	D2 B5	setb		p3.3	;enable=1
0094	75 F0 FF	mov		b,#0ffh	;small delay
0097	D5 F0 FD	djnz		b,$	
009A	C2 B5	clr		p3.3	;enable=0
009C	75 F0 FF	mov		b,#0ffh	;small delay
009F	D5 F0 FD	djnz		b,$	
00A2	22	ret			;return to main program
00A3				t_display:	
00A3	74 54	mov		a,#54h	;code for 'T'
00A5	12 00 8E	lcall		data_disp	;display it on LCD module
00A8	74 5F	mov		a,#5fh	;code for '_'
00AA	12 00 8E	lcall		data_disp	;display it on LCD module
00AD	74 6F	mov		a,#6fh	;code for 'o'
00AF	12 00 8E	lcall		data_disp	;display it on LCD module
00B2	74 6E	mov		a,#6eh	;code for 'n'
00B4	12 00 8E	lcall		data_disp	;display it on LCD module
00B7	74 3D	mov		a,#3dh	;code for '='
00B9	12 00 8E	lcall		data_disp	;display it on LCD module
00BC	22	ret			;return to main program
00BD				display:	

Program 30 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
00BD	EC	mov		a,r4	;read most significant digit ;of ON time from r4
00BE	44 30	orl		a,#30h	;convert it into ASCII
00C0	12 00 8E	lcall		data_disp	;display it on LCD
00C3	74 2E	mov		a,#2eh	;load acc. with code for '.'
00C5	12 00 8E	lcall		data_disp	;display it on LCD
00C8	ED	mov		a,r5	;read next digit
00C9	44 30	orl		a,#30h	;convert it into ASCII
00CB	12 00 8E	lcall		data_disp	;display it on LCD
00CE	EE	mov		a,r6	;read next digit
00CF	44 30	orl		a,#30h	;convert it into ASCII
00D1	12 00 8E	lcall		data_disp	;display it on LCD
00D4	EF	mov		a,r7	;read next digit
00D5	44 30	orl		a,#30h	;convert it into ASCII
00D7	12 00 8E	lcall		data_disp	;display it on LCD
00DA	22	ret			;return to main program ;delay program
00DB				delay:	
00DB	7D 05	mov		r5,#05h	
00DD	7F FF	l8: mov		r7,#0ffh	
00DF	7E FF	l7: mov		r6,#0ffh	
00E1	DE FE	djnz		r6,$	
00E3	DF FA	djnz		r7,l7	
00E5	DD F6	djnz		r5,l8	
00E7	22	ret			
					;input 24 bit hex number is in r2_hb,r1,r0_lb ;output bcd number is in r0_hb,r1,r2,r3,r4,r5,r6,r7_lb
00E8				hex_bcd:	
00E8					
00E8	7F 19	mov		r7,#19h	
00EA	12 01 49	lcall		conv1	
00ED	EB	mov		a,r3	
00EE	FC	mov		r4,a	
00EF	7F 19	mov		r7,#19h	
00F1	12 01 49	lcall		conv1	
00F4	EB	mov		a,r3	
00F5	FD	mov		r5,a	
00F6	7F 19	mov		r7,#19h	
00F8	12 01 49	lcall		conv1	
00FB	EB	mov		a,r3	
00FC	FE	mov		r6,a	
00FD	7F 19	mov		r7,#19h	
00FF	12 01 49	lcall		conv1	
0102	EB	mov		a,r3	
0103	FF	mov		r7,a	

Program 30 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0104	EC	mov		a,r4	
0105	54 0F	anl		a,#0fh	
0107	F8	mov		r0,a	
0108	EC	mov		a,r4	
0109	54 F0	anl		a,#0f0h	
010B	C4	swap		a	
010C	F9	mov		r1,a	
010D	ED	mov		a,r5	
010E	54 0F	anl		a,#0fh	
0110	FA	mov		r2,a	
0111	ED	mov		a,r5	
0112	54 F0	anl		a,#0f0h	
0114	C4	swap		a	
0115	FB	mov		r3,a	
0116	EE	mov		a,r6	
0117	54 0F	anl		a,#0fh	
0119	FC	mov		r4,a	
011A	EE	mov		a,r6	
011B	54 F0	anl		a,#0f0h	
011D	C4	swap		a	
011E	FD	mov		r5,a	
011F	EF	mov		a,r7	
0120	54 0F	anl		a,#0fh	
0122	FE	mov		r6,a	
0123	EF	mov		a,r7	
0124	54 F0	anl		a,#0f0h	
0126	C4	swap		a	
0127	FF	mov		r7,a	
0128	EF	mov		a,r7	
0129	F5 08	mov		08h,a	
012B	EE	mov		a,r6	
012C	F5 09	mov		09h,a	
012E	ED	mov		a,r5	
012F	F5 0A	mov		0ah,a	
0131	EC	mov		a,r4	
0132	F5 0B	mov		0bh,a	
0134	E8	mov		a,r0	
0135	FF	mov		r7,a	
0136	E9	mov		a,r1	
0137	FE	mov		r6,a	
0138	EA	mov		a,r2	
0139	FD	mov		r5,a	
013A	EB	mov		a,r3	
013B	FC	mov		r4,a	
013C	E5 08	mov		a,08h	
013E	F8	mov		r0,a	
013F	E5 09	mov		a,09h	
0141	F9	mov		r1,a	

Program 30 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0142	E5 0A	mov		a,0ah	
0144	FA	mov		r2,a	
0145	E5 0B	mov		a,0bh	
0147	FB	mov		r3,a	
0148					
0148	22	ret			
0149	C3	conv1: clr		c	
014A	74 00	mov		a,#00h	
014C	FB	mov		r3,a	
014D					
014D	C3	conv2: clr		c	
014E	74 00	mov		a,#00h	
0150	DF 01	djnz		r7, cont	
0152	22	ret			
0153				cont:	
0153	E8	mov		a,r0	
0154	33	rlc		a	
0155	F8	mov		r0,a	
0156	E9	mov		a,r1	
0157	33	rlc		a	
0158	F9	mov		r1,a	
0159	EA	mov		a,r2	
015A	33	rlc		a	
015B	FA	mov		r2,a	
015C	EB	mov		a,r3	
015D	F5 F0	mov		b,a	
015F	35 F0	addc		a,b	
0161	D4	da		a	
0162	FB	mov		r3,a	
0163	50 E8	jnc		conv2	
0165	E8	mov		a,r0	
0166	34 00	addc		a,#00h	
0168	F8	mov		r0,a	
0169	E9	mov		a,r1	
016A	34 00	addc		a,#00h	
016C	F9	mov		r1,a	
016D	EA	mov		a,r2	
016E	34 00	addc		a,#00h	
0170	FA	mov		r2,a	
0171	02 01 4D	ljmp		conv2	
0174		end			

```
/* program for pulse width measurement using INT0 interrupt and
timer0*/
#include <stdio.h>
#include <reg51.h>

   sbit   rs = P3^2;
   sbit    e = P3^3;


   int d,d1,d2,d3,d4,d5,v;
   float v1;

   void main (void)
   {
        void lcd_int(void);
   void lcd_comand(void);
   void lcd_data(void);
   void data_conv(void);
   void display(void);
   int i, hc,lc,tp;

   int c[] = {'T','o','n','='};
   lcd_int();           //initialization LCD module
   F0 = 0;              //clear user flag
   TMOD = 0x09;         //timer-0 16-bit timer
   TH0 = 0x00;          //clear timer-0
   TL0 = 0x00;
   TCON = 0x11;         //start timer-0
                        //external-0 interrupt is
                        //transition activated
   IE = 0x81            //enable external-0 interrupt
                        //enable global interrupt enable bit
   while(1)
   {
   while(F0 == 0);      //stay in loop until F0 goes high
   F0 = 0;              //Clear F0 flag

   lc = TL0;            //read low byte timer-0
   hc = TH0;            //read high byte of timer-0
   tp = hc*256+lc;           //assign 16-bit number to 'tp'
   data_conv();         //unpack 16-bit number

   P1 = 0x02;           //bring cursor to left corner of display
   lcd_comand();        //send to display
```

```
   for (i = 1;i<5;i++)       //display 'Ton=' on LCD module
   {
   P1 = c[i];
   lcd_data();
   }
        display();           //display ON Time in ms

        while(INT0 == 0);    //start measurement at low to high
        while(INT0 != 0);    //transition of the input signal
   TMOD = 0x09;
    TH0 = 0x00;
    TL0 = 0x00;
    TCON = 0x11;
    IE   = 0x81;
    }
    }
   void p_width_isr(void) interrupt 0
   {
   IE = 0x00;    //disable all interrupts
   TCON = 0x00;  //stop timer-0
   F0   = 1;     //set user flag
   }
 /* subroutine to separate digits */
   void data_conv(void)
   {
   v = v*10000.0;
   d = v;
   d1 = d/10000;

   d2 = d % 10000;
   d2 = d2/1000;

   d3 = d % 1000;
   d3 = d3/100;

   d4 = d % 100;
   d4 = d4/10;

   d5 = d % 100;
   d5 = d5/10;
   }
/* subroutine to display ON-time on LCD module */
   void display(void)
```

```
   {
   d1 = d1+0x30;
   P1 = d1;
   lcd_data();

   P1 = 0x2e;
   lcd_data();

   d2 = d2+0x30;
   P1 = d2;
   lcd_data();

   d3 = d3+0x30;
   P1 = d3;
   lcd_data();

   d4 = d4+0x30;
   P1 = d4;
   lcd_data();

   d5 = d5+0x30;
   P1 = d5;
   lcd_data()
        }

   /* LCD initialization subroutine */
void lcd_int(void)
   {
   P1 = 0x38;
   lcd_comand();

   P1 = 0x0e;
   lcd_comand();

   P1 = 0x06;
   lcd_comand();

   P1 = 0x01;
   lcd_comand();

   P1 = 0x02;
   lcd_comand();
   }
```

```
/* subroutine for sending control words to LCD module */
  void lcd_comand(void)
  {
  int i;
  rs = 0;
  e =  1;
  for (i = 1;i<1000;i++);
  e = 0;
  for (i = 1;i<1000;i++);
  }
  /* subroutine for sending data to display on LCD module */
  void lcd_data(void)
  {
  int i;
  rs = 1;
  e =  1;
  for (i = 1;i<1000;i++);
  e = 0;
  for (i = 1;i<1000;i++);
  }
```

Experiment 27

AIM: To measure phase angle between two signals

COMPONENTS: Microcontroller 89C51, 12.0MHz Quartz crystal, 8.2kΩ, 100Ω, resistors, switch, 20kΩ multi-turn pot, 0.1μF decoupling capacitors, LCD module, bread board and connecting wires.

EQUIPMENT: Power supply +5V/1A.

Principle: The principle involved in the design of microcontroller based phase meter is similar to the principle involved in the conventional digital phase meter. The measurement of phase is based on the relationship of the phase angle to the delay between the zero crossings of two input signals. As shown in Fig. 30 phase shift 'ϕ' and the delay 'τ' between the zero crossings of the signals by a simple equation presented below:

$$\Phi = (\tau/T)^* 360 \qquad \ldots\ldots\ldots\ldots (1)$$

where 'T' is the time period of the signal and 'Φ' is phase difference in degrees.

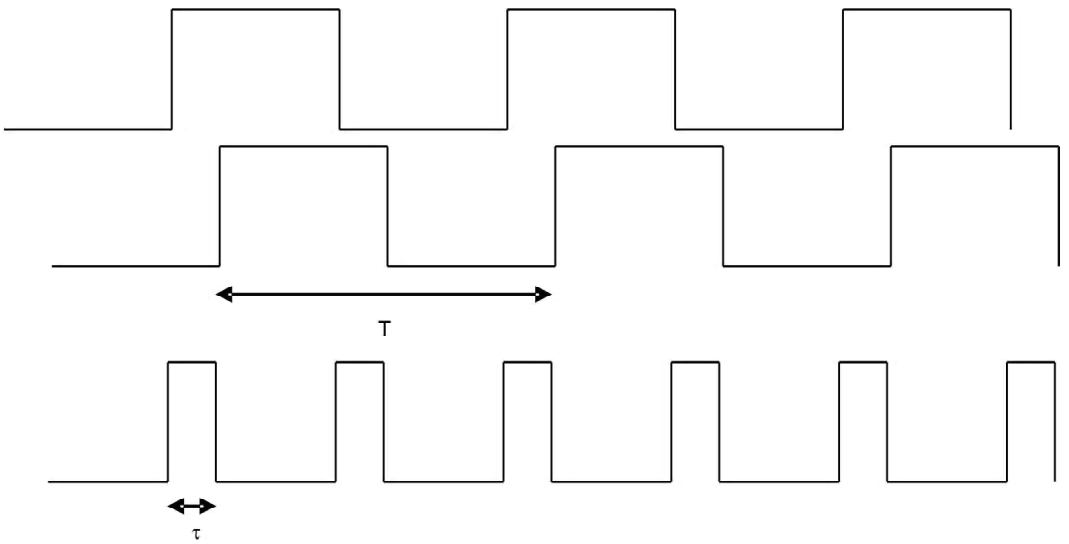


Fig. 30. Phase shift 'ϕ' and the delay 'τ' between the zero crossings of the signals.

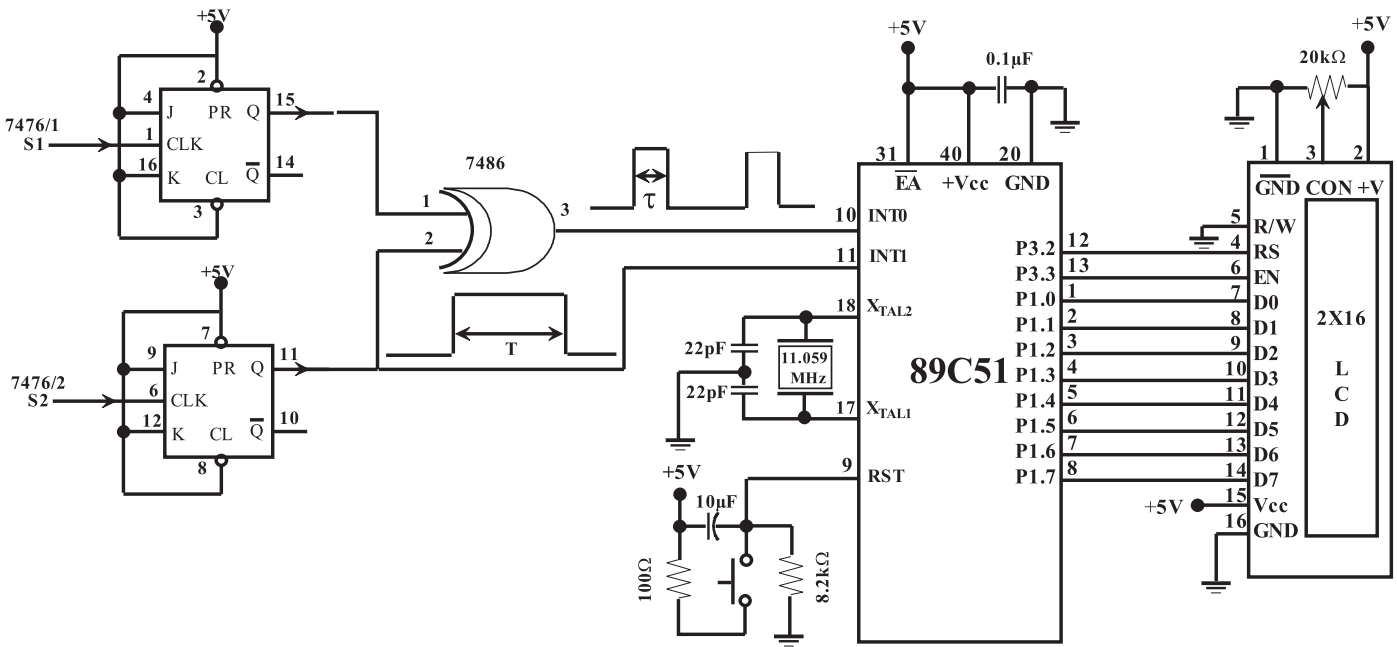


Fig. 31. The schematic diagram of microcontroller based phase meter.

Hardware: Figure 31 shows the schematic of microcontroller based phase meter. The flip-flops A and B convert the input signals S1 and S2 into square wave signals of 50% duty cycle. The frequency of the output of flip-flops will be half that of the input signals' frequency. The outputs of these two flip-flops are applied to an XOR gate which produces an output whose 'ON' time is proportional to the phase difference between these two signals. The 'ON' time of the XOR gate linearly varies with the phase from 0-360°. Figure 32 shows the linear relationship with the phase angle and output of XOR gate. If the two signals are directly applied to the XOR gate, only 180° phase difference can be measured.

The output of the XOR gate i.e., 't' and the output of the reference flip-flop i.e. 'T' are applied to the inputs INTO and INT1 of 89C51 respectively. Here timer-0 and timer-1 are used to measure the "ON' time of the signals applied at the inputs INT0 and INT1. The timer-0 and timer-1 are programmed in mode-1 and GATE bits in TMOD register are set. This makes the internal 1MHz clock to be available to timer-0 and timer-1 for counting only when INT0/INT1 pins are HIGH and TR0/ TR1 in TCON register are set. The interrupt is programmed to occur on a HIGH-to-LOW edge on INT0/INT1 pins. Initially timer-0/timer-1 are cleared and TR0/TR1 bits are set, the timer-0/timer-1 begins counting when the input pulse goes HIGH, and stops when the pulse goes from HIGH-to-LOW transition. This HIGH-to-LOW transition of the pulse also generates interrupt signal to the microcontroller. The program control will be transferred to the interrupt service routine and reads the contents of timer-0/timer-1. When the interrupts are generated the timers content will be read and substituted in the equation (1) to get the phase angle between two signals in degrees. The phase is then displayed on LCD module which is interfaced with the microcontroller.

Software: The software first initializes stack pointer and LCD module. After initialization of LCD module, the program then initializes timer-0 16-bit timer in mode-1 and GATE bit in TMOD register is set. Then timer-0 is loaded with 0000H, then software sets TR0 bit in TCON register. When the INT0 input goes HIGH then timer-0 starts counting internal clock (1 MHz). The program then enables timer-0 interrupt by setting ET0 and EA in interrupt enable register. The external pulse HIGH-to-LOW transition stops the timer-0 and generates an interrupt signal which will interrupt CPU and program control will be transferred to the location 0003H from there program control again transferred to the location 'tow_isr', there it disables interrupts and reads the contents of timer-0. Then the count is assigned to variable 'tow'. The same procedure is repeated for measurement of 'T' using timer-1 and INT1 input. Finally the program calculates the phase angle by substituting the 't' and 'T' in the equation (1). For continuous measurement, the program re-initializes the timer-0/timer-1 and enables interrupts and then returns to main program.

a. Initialize stack pointer with the address 40H
b. Initialize LCD module
c. Call subroutine to display 'Phase=' on LCD module
d. Initialize timer-0 as 16-bit timer by sending Mode word (09H)

e. Clear timer-0 for measuring ON time of TTL signal in terms of μsec.
f. Set timer-0 interrupt ET0 & EA bits
g. Enter into infinite loop
h. When timer-0 interrupt occurs, then read contents of timer-0 and store it as 't'
i. Initialize timer-1 as 16-bit timer by sending mode word (90H)
j. Clear timer-1 for measuring ON time of TTL signal in terms of μsec.
k. Set timer-0 interrupt ET1 & EA bits
l. Enter into infinite loop
m. When timer-1 interrupt occurs, then read contents of timer-1 and store it as 'T'
n. Calculate phase angle by substituting 't' and 'T' in the equation
o. Display phase angle on LCD module
p. Poll INT0/INT1 inputs until they go HIGH
q. Poll INTO/INT1 inputs until they go LOW to start measurement exactly from LOW to HIGH transition
r. Initialize timer-0 and timer-1 and enable their interrupts
s. Return to main program.

Procedure

1. Wire the circuit as shown in Fig. 31.
2. Load the assembly language program–31 (embedded "C' program) into the program memory from the starting address 0000H using 89C51 programmer.
3. Execute the program by supplying + 5V power supply to the circuit.
4. Observe the display of phase angle of the input signals on LCD module.
5. Vary the phase angle using phase shifting networks and record the corresponding data displayed on LCD module.
6. Tabulate the results in Table 19.
7. Compare this data with oscilloscope readings

Table 19

S.No.	Phase Oscilloscope	Phase displayed on LCD module
1	0	0.0
2	45	45.2
3	135	134.8
4	180	180.4
5	315	315.0

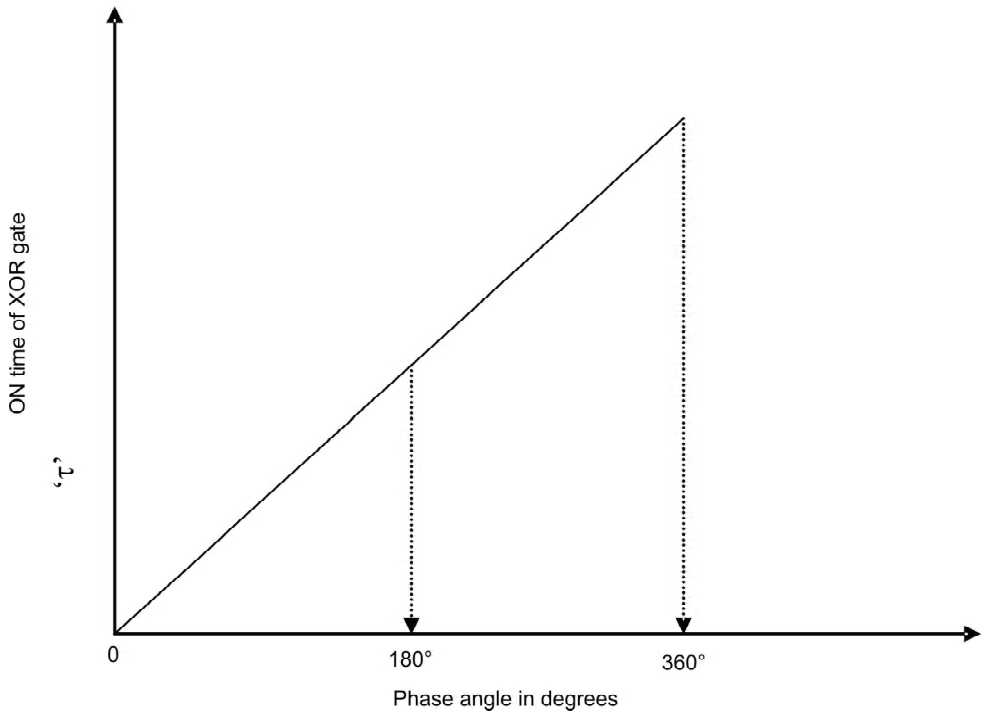


Fig. 32. The linear relationship between the phase angle and output of XOR gate.

```
/* Program-31: program for phase measurement using INT0:INT1 and
Timer0:Timer1 */

#include <reg51.h>

sbit   rs = P3^2;
sbit    e = P3^3;
bit flag1;
bit flag2;

void  lcd_int(void);
void  lcd_comand(void);
void  lcd_data(void);
void  display(void);

unsigned int d,d1,d2,d3,d4,d5;

void main(void)
{
```

```
    unsigned int tow,T;
    int i;
    float phase;
    unsigned int p;
   int m[] = {'P','h','a','s','e','='};

    lcd_int(); //initialization LCD module

   while(1)
   {
   while(INT0 == 0);
   while(INT0 == 1);

   TMOD = 0x09;  //timer-0 16-bit timer

   TH0   = 0x00; //clear timer-0
   TL0   = 0x00;

   TCON  = 0x11; //start timer-0
//transition activated
//enable external-0 interrupt

   IE = 0x81; //external-0 interrupt is
//enable global interrupt enable bit

   while(flag1==0); //stay in loop until flag1 sets
   flag1 = 0;    //clear flag

   tow = (TH0*256)+TL0;    //read low byte timer-0
                                        //read high byte of timer-
0
    //assign 16-bit number to 'tow'

   while(INT1 == 0);    //start measurement at low to high
   while(INT1 == 1);    //transition of the input signal

   TMOD = 0x90;     //timer-1 16-bit timer
   TH1   = 0x00;    //clear timer-1
   TL1   = 0x00;

   TCON  = 0x44;    //start timer-0
   //transition activated
   //enable external-0 interrupt
```

```
   IE = 0x84;  //external-0 interrupt is
 //enable global interrupt enable bit


      while(flag2 == 0); //stay in loop until flag2 sets
   flag2 = 0;      //clear flag

   T = (TH1*256)+TL1;   //read low byte timer-0
                                  //read high byte of timer-0
            //assign 16-bit number to 'tow'

   phase = (tow*36000)/T; //calculate phase
   p = (int)phase;          //convert into integer

   d1 = p/10000;        //convert into unpacked number
   d2 = p % 10000;      //to display on LCD module

   d2 = d2/1000;
   d3 = p % 1000;

   d3 = d3/100;
   d4 = p % 100;

       d4 = d4/10;
   d5 = p % 10;

      for (i=0; i<6; i++)    //Display 'Phase=' on LCD module
   {
    P1 = m[i];
    lcd_data();
   }

      display();     //Subroutine to display phase in degrees

       P1 = 0xdf;
      lcd_data();

       P1 = 0x02;
      lcd_comand();
   }
   }
/* ISR to clear TR0,IT0, EX0, EA and to set flag=1 */
```

```
    void tow_isr(void) interrupt 0
   {
   IE = 0x00;
   TCON = 0x00;
   flag1 = 1;
   }
/* ISR to clear TR1,IT1, EX1, EA and to set flag=1 */

   void T_isr(void) interrupt 2
   {
   IE = 0x00;
   TCON = 0x00;
   flag2 = 1;
   }
      void display(void)
   {
   d1 = d1+0x30;
   P1 = d1;
   lcd_data();

   d2 = d2+0x30;
   P1 = d2;
   lcd_data();

   d3 = d3+0x30;
   P1 = d3;
   lcd_data();

   P1 = 0x2e;
   lcd_data();

   d4 = d4+0x30;
   P1 = d4;
   lcd_data();
       }

/* LCD initialization subroutine */
void lcd_int(void)
  {
  P1 = 0x38;
  lcd_comand();

  P1 = 0x0e;
  lcd_comand();
```

```
 P1 = 0x06;
 lcd_comand();

 P1 = 0x01;
 lcd_comand();

 P1 = 0x02;
 lcd_comand();
 }
/* subroutine for sending control words to LCD module */
void lcd_comand(void)
 {
 int i;
 rs = 0;
  e = 1;
 for (i = 1;i<9000;i++);
  e = 0;
 for (i = 1;i<9000;i++); }
/* subroutine for sending data to display on LCD module */

void lcd_data(void)
 {
  int i;
  rs = 1;
   e = 1;
   for (i = 1;i<9000;i++);
   e = 0;
   for (i = 1;i<9000;i++);
 }
```

Experiment 28

AIM: To interface MAX232 with microcontroller and transmit data to and from the personal computer.

COMPONENTS: 89C51, MAX232, 12.0 MHz crystal, switch, LCD module, 1.0μF-4, 0.1μF-2 8.2kΩ resistor, 100Ω resistor, switch, bread board, and connecting wires.

EQUIPMENT: Personal computer.

INTRODUCTION: The serial port provides connection between the external designed system to a personal computer for transmission, storage, and further analysis of the acquired data. The IC MAX232 can be used for serial communication with the host computer. This enables the user to store the information, in the personal computer, regarding the parameter being measured through this port. This facility makes the instrument suitable for research application. The MAX232 line driver/receiver is intended for all serial communications interfaces, particularly applications where ±12V is not available. These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5μW. The MAX232 uses minimum external components.

Hardware Description : MAX232, a RS232 line driver/receiver, is interfaced to AT89C51 microcontroller as shown in the Fig. 33. The TxD (Transmit Data) and RxD (Receive Data) pins of microcontroller are connected to $T1_{IN}$ and $R1_{OUT}$ of MAX232 respectively for serial data transmission and reception. The port line P3.2 of microcontroller is used as DTR (Data Terminal Ready) control signal and is connected to $T2_{IN}$ pin of MAX232. Similarly, the port line P3.3 of microcontroller is used as DSR (Data Set Ready) control signal and is connected to $R2_{OUT}$ pin of MAX232. The corresponding $T1_{OUT}$, $T2_{OUT}$, $R1_{IN}$ and $R2_{IN}$ of MAX232 are terminated on a D9 female connector for attaching an external system to the PC. The circuit diagram also presents the interfacing of the LCD display module with the microcontroller. The data bus D0-D7 of display module is connected to P1.0-P1.7 of microcontroller and the control pins RS and E are connected to the P3.4 and P3.5 pins of 89C51 respectively.

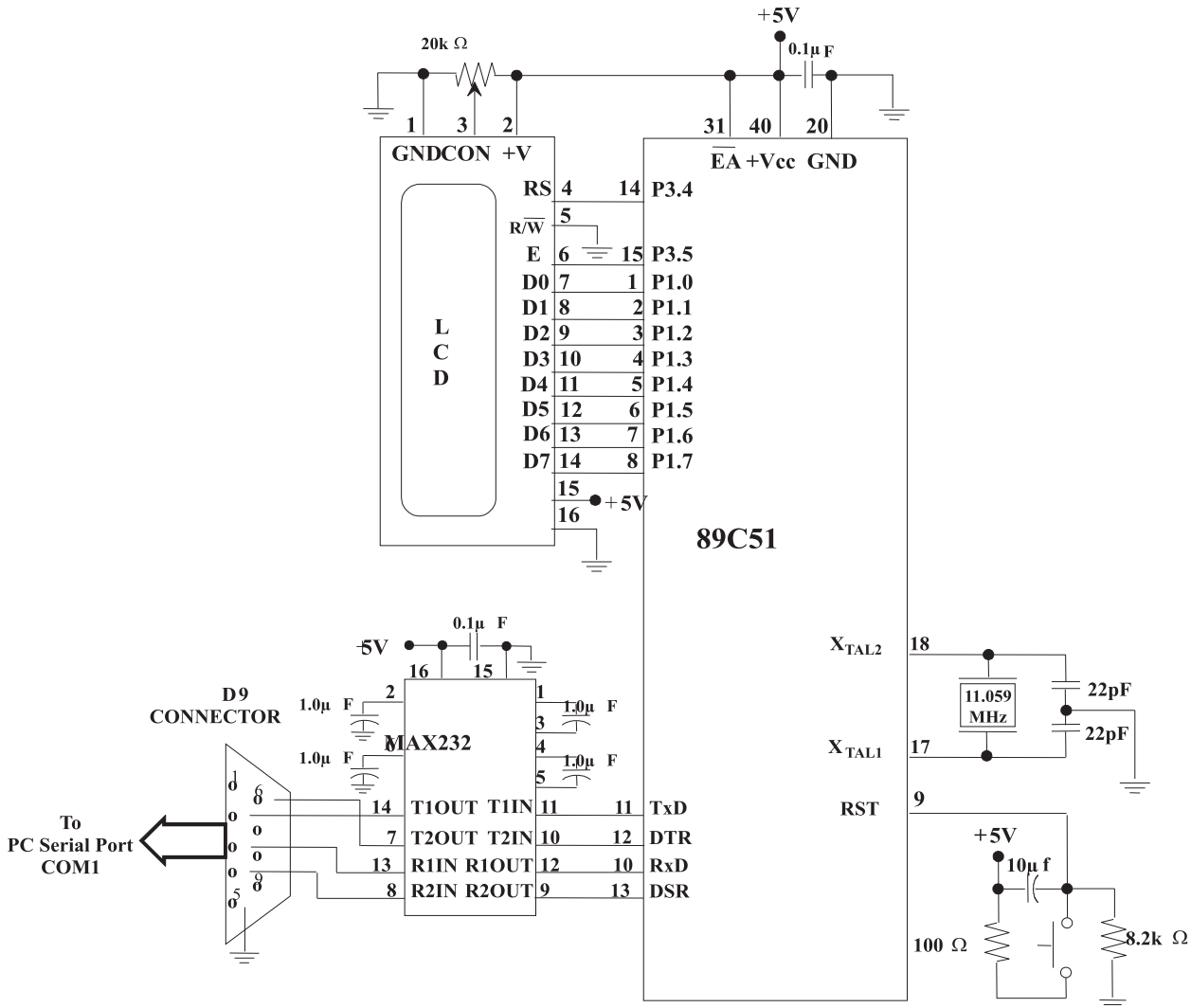


Fig. 33. Interfacing of MAX232 with microcontroller.

Software Description: The software initializes LCD module, the program then initializes timer-0 as 16-bit timer in mode-1 in mode-2 (auto-reload mode) to generate baud rate clock for on-chip serial port of the microcontroller. Then it initializes serial port in mode-1 with 9600 baud rate. Then label 'INSTRUMENTATION' is sent character by character to the serial port. The serial data receiving program is written 'C' language. The program first initializes the serial port of the computer with 9600 baud rate. The program then receives the data from microcontroller through serial port. The 'C' program finally displays the data received on its monitor.

Program for receiving data from microcontroller and displaying on monitor

First 'C' program-32 should be compiled and executed first on personal computer. The program waits for serial data, then the embedded 'C' program-33 should be run on microcontroller. Then, the label 'INSTRUMENTATION' will be displayed on the monitor.

Procedure

1. Wire the circuit as shown in Fig. 33.
2. Connect the microcontroller serial port to personal computer serial port through MAX232.
3. Run 'C' program-32 on personal computer.
4. Load the embedded "C' program-33 into the program memory from the starting address 0000H using 89C51 programmer.
5. Execute the program by supplying +5V power supply to the circuit.
6. Observe the display label 'INSTRUMENTATION' on the monitor of the personal computer.

Program for transmitting data to microcontroller for displaying on LCD module

First the embedded 'C' program-35 should be executed on microcontroller, and then it prompts cursor on LCD module and waits for data from personal computer. Then run 'C' program-34 on personal computer, it sends the label 'INSTRUMENTATION' to microcontroller through its serial port. The microcontroller receives the data and displays it on LCD module.

Procedure

1. Wire the circuit as shown in Fig. 33.
2. Connect the microcontroller serial port to personal computer serial port through MAX232.

3. Load the embedded "C' program-35 into the program memory from the starting address 0000H using 89C51 programmer and run it.
4. Run 'C' program-34 on personal computer.
5. Observe the display label 'INSTRUMENTATION' on LCD module.

Program-32:

```
/* C program to read data from microcontroller through serial
port.
    This program should be executed in the personal computer*/

#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<bios.h>

void main(void)
   {
   int i;
   char x[15];
   clrscr();4
   bioscom(0,0xe3,0);
   for(i=0; i<15; i++)
   {
   While(bioscom(3,0,0) & 0x0020)==0); //wait for DSR high
   While(bioscom(3,0,0) & 0x0100)==0);
   x[i] = inport(0x3f8);
   }
   for(i=0; i<15; i++)
   printf("%c",x[i]);
   }
```

Program-33:

```
/* Embedded 'C' program to send data to personal computer through
    serial port for displaying a label 'INSTRUMENTATION' on personal
    computer monitor. This program should be executed in the
    microcontroller */
#include <reg51.h>
#include<stdio.h>
sbit    dtr = P3^2;
sbit    dsr = P3^3;
```

```
void main(void)
   {
   int i,j;
   void ser_out(void);
   char  data1[15] = "INSTRUMENTATION";
   TMOD = 0x20;
   TCON = 0x00;
   TH1   = 0xFD;
   SCON = 0x40;
   TI    = 1;
   TR1   = 1;
   for(i = 0; i<15; i++)
   {
   dtr = 1;
   while(TI != 1);
   TI = 0;
   SBUF = data1[i];
   dtr = 0;
   }
while(1);
   }
```

Program-35:

```
/* C program to send data to microcontroller through serial port.
    This program should be executed in the personal computer*/

#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<bios.h>

void main()
   {
        int s,i;
        char   x[15] ="INSTRUMENTATION";
        clrscr();

        bioscom(0,0xe3,0);
        for(i = 0; i<15; i++)
        {
        outportb(0x3fc,0x10);                        //DTR is high
        while((bioscom(3,0,0) & 0x2000) == 0);
        outportb(0x3f8,x[i]);                        //DTR is low
        outportb(0x3fc,0x00);
```

```
        }
        for(i = 0; i<15; i++)
        printf("%c",x[i]);
        }
```

Program-35:

```
/* Embedded 'C' program to receive data from personal computer
through
    serial port for displaying a label 'INSTRUMENTATION' on LCD
module.
    This program should be executed in the microcontroller */

#include <reg51.h>
#include<stdio.h>
sbit  rs = P3^4;
sbit    e = P3^5;
sbit    dtr = P3^2;
sbit    dsr = P3^3;

void main(void)
   {
    void   lcd_init(void);
    void   lcd_comand(void);
    void   lcd_data(void);
    void   display(void);

   int i,j;
   char   x[15];

   TMOD = 0x20;
   TCON = 0x00;
   TH1   = 0xFD;
   SCON = 0x40;
   RI    = 1;
   TR1   = 1;
   lcd_init();
   for(i = 0; i<15; i++)
         {
   REN = 1;
   while(dsr == 0);
   while(RI != 1);
   RI = 0;
   x[i] = SBUF;
```

```
}

     for(i=1;i<16;i++)
      {
      P1 = x[i];
     lcd_data();
      }
While(1);
}
void lcd_init(void)
      {
 P1 = 0x38;
 lcd_comand();
       P1 = 0x0e;
 lcd_comand();
 P1 = 0x06;
 lcd_comand();
       P1 = 0x01;
      lcd_comand();
       P1 = 0x02;
 lcd_comand();
     }

void lcd_comand(void)
{
   int i;
   rs = 0;
   e = 1;
   for (i = 1;i<9000;i++);
   e = 0;
   for (i = 1;i<9000;i++);
   }

void lcd_data(void)
   {
         int i;
         rs = 1;
            e = 1;
         for (i = 1;i<9000;i++);
            e = 0;
         for (i = 1;i<9000;i++);
         }
```

PART III
Projects

1

Microcontroller Based pH Measurement System

AIM: To design and construct microcontroller based pH measurement system.

COMPONENTS: Microcontroller 89C51, glass electrode, Op-amp OPA111, 2-Op-amp 3140, A/D converter ICL7135, precision voltage regulator LM329, 2-5.1V zener diodes, 0.1μF, 0.47μF, 1.0μF (metalized polyester capacitors), 3-10 kΩ, 1-100 kΩ multi-turn pots, 2–100kΩ, 1kΩ and 27Ω resistors, LCD module, bread board and connecting wires.

EQUIPMENT: Power supply + 5V/3A, ± 12V/100mA.

INTRODUCTION: The degree of acidity or alkalinity of a solution is expressed by pH scale. The pH scale is defined as the negative logarithm of hydrogen ion concentration as given below.

$$pH = -\log_{10} [H^+]$$

In a solution if hydrogen ions H^+ are more, then solution shows acidic characteristics and for more hydroxyl ions OH^-, solution shows basic characteristics. In pure water they are equal in number hence, it is called neutral solution.

Hardware: Figure 34 shows the complete schematic diagram of microcontroller based pH measurement system. An industrial standard glass electrode is employed for the present experiment. It consists of reference and measuring electrode assembled in a single glass tube. The electrode consists of silver rod which is bonded to a glass membrane and surrounded by a potassium chloride solution. The electrode works on principle that when a glass surface is kept in contact with the solution a potential

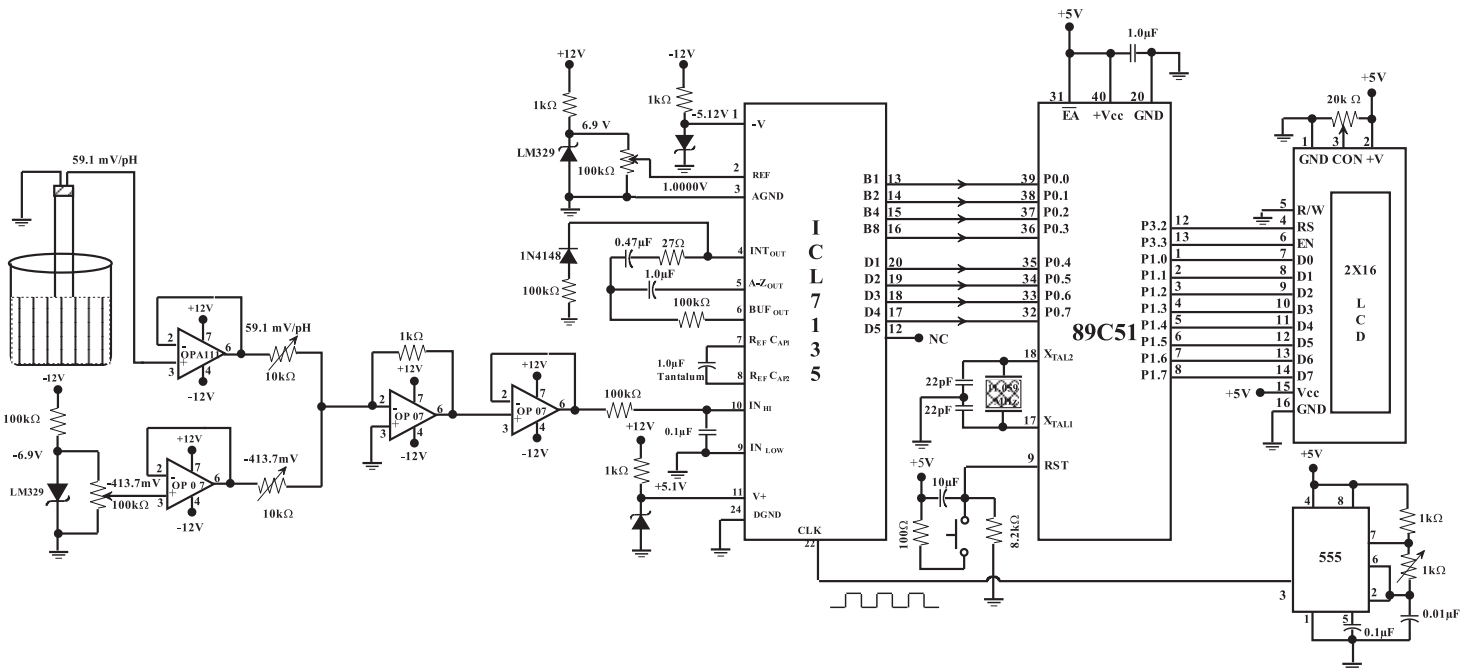


Fig. 34. Schematic of microcontroller based pH measurement system.

will be established between glass surface and solution and this potential difference is a function of hydrogen ion concentration of solution. The glass electrode produces + 413.7 mV for 0.0 pH, 0.0 mV for pH 7.0 and – 413.7mV for 14.0 pH, and a voltage change of 59.1mV for every 1.0 pH at 25°C.

The circuit diagram also shows the signal conditioning circuit for measurement of pH of solution. It consists of buffer and summing amplifier. The glass electrode produces every small current of the order of nano amperes; hence, the measuring circuit should have high impedance. The operational amplifier A1 (OPA111) is used in voltage follower configuration to avoid loading effect. The summing amplifier is an inverting amplifier with two inputs, one from pH buffer amplifier A1 and other from voltage follower A2 which provides – 413.7 mV. The output of operational amplifier A3 is given by the following equation.

$$V_{out} = (R_f/R) * (V_m + V_{ref})$$

Where R_f = 1 kΩ, R = 5.91kΩ. Hence, the output of amplifier A3 is ±10 mV/pH and same will be given to the A/D converter. Figure also shows the interfacing A/D converter and LCD module with microcontroller, the port-0 of microcontroller is used to read the digit strobes and BCD data from A/D converter and port-1 of microcontroller is used to interface LCD module. The microcontroller reads the digital data from A/D converter and displays the same on LCD module.

Software: The software first initializes the stack pointer and the LCD module. After initialization the program then calls the A/D conversion subroutine to read the voltage of the pH sensor through analog to digital converter ICL7135. This digital data from A/D converter directly indicates the pH of solution, same will be sent to the LCD module. The assembly program is given in program-36 and embedded C program in given in program-37. The execution of the program displays the pH of a solution on LCD module.

a. Initialize stack pointer
b. Initialize LCD module
c. Call subroutine to display 'pH=' on LCD module
d. Call subroutine to convert analog voltage into digital data (BCD)
e. Call subroutine to display ADC output on LCD module
f. Send control word to the LCD module to bring the cursor to the left corner of the module
g. Repeat the above steps from (c).

Table 20

S.No.	pH of solution with commercial	pH of solution with present system Digital pH meter
1	4.0 (pH Buffer 4)	4.00
2	9.0 (pH Buffer 9)	9.00

Program 36 : Program for measurement of pH of a solution

Address	Opcode	Label	Mnemonic	Operand	Comments
		org	0000h		
					;Initialization of LCD display
0000	75 81 40	mov		sp,#40h	;Initialize stack pointer ;with the starting address 40h
0003	74 38	mov		a,#38h	;function set control word
0005	12 00 5A	lcall		enable	;send to LCD module and
0008	74 0E	mov		a,#0eh	;display ON/OFF control word
000A	12 00 5A	lcall		enable	
000D	74 06	mov		a,#06h	;entry mode set control word
000F	12 00 5A	lcall		enable	
0012	74 01	mov		a,#01h	;clear display control word
0014	12 00 5A	lcall		enable	
0017	74 02	mov		a,#02h	;return home control word
0019	12 00 5A	lcall		enable	
001C	12 00 84	lcall		pH_disp	;call subroutine to display ;'pH=' on LCD module
001F	12 00 2D	lcall		adc	;call subroutine A/D conversion
0022	12 00 94	lcall		display	;call subroutine to display ;to display pH
0025	74 02	mov		a,#02h	;bring cursor to left
0027	12 00 5A	lcall		enable	;corner of the display
002A	02 00 1C	ljmp		l1	;repeat the program ;program for ICL7135 A/D converter
002D				adc:	
002D	E5 80	x0: mov		a,p0	;read digit strobes and ;data from A/D converter
002F	54 F0	anl		a,#0f0h	;mask data
0031	B4 00 F9	cjne		a,#00h,x0	;check whether D1-D4 digit ;strobes are LOW, if not ;go to location x0
0034	E5 80	mov		a,p0	;if all digit strobes are ;LOW, the read data from ;A/D converter
0036	54 0F	anl		a,#0fh	;mask digit strobes
0038	F8	mov		r0,a	;store MSD data in r0
0039	30 84 FD	jnb		p0.4,$	;poll digit strobe D4 until ;it goes HIGH
003C	E5 80	mov		a,p0	;read digital data
003E	54 F0	anl		a,#0f0h	;mask digit strobes
0040	F9	mov		r1,a	;store data in r1

Program 36 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0041	30 85 FD	jnb		p0.5,$	;poll digit strobe D3 until ;it goes HIGH
0044	E5 80	mov		a,p0	;read digital data
0046	54 0F	anl		a,#0fh	;mask digit strobes
0048	FA	mov		r2,a	;store data in r2
0049	30 86 FD	jnb		p0.6,$	;poll digit strobe D2 until ;it goes HIGH
004C	E5 80	mov		a,p0	;read digital data
004E	54 0F	anl		a,#0fh	;mask digit strobes
0050	FB	mov		r3,a	;store data in r3
0051	30 87 FD	jnb		p0.7,$	;poll digit strobe D1 until ;it goes HIGH
0054	E5 80	mov		a,p0	;read digital data
0056	54 0F	anl		a,#0fh	;mask digit strobes
0058	FC	mov		r4,a	;store data in r4
0059	22	ret			;return to main program
				enable:	
005A	F5 90	mov		p1,a	;send control word to display
005C	C2 B4	clr		p3.2	;RS=0
005E	D2 B5	setb		p3.3	;enable=1
0060	75 F0 FF	mov		b,#0ffh	;small delay
0063	D5 F0 FD	djnz		b,$	
0066	C2 B5	clr		p3.3	;enable=0
0068	75 F0 FF	mov		b,#0ffh	;small delay
006B	D5 F0 FD	djnz		b,$	
006E	22	ret			;return to main program
				data_disp:	
006F	F5 90	mov		p1,a	;send character code to display
0071	D2 B4	setb		p3.2	;RS=1
0073	D2 B5	setb		p3.3	;enable=1
0075	75 F0 FF	mov		b,#0ffh	;small delay
0078	D5 F0 FD	djnz		b,$	
007B	C2 B5	clr		p3.3	;enable=0
007D	75 F0 FF	mov		b,#0ffh	;small delay
0080	D5 F0 FD	djnz		b,$	
0083	22	ret			;return to main program
				pH_disp:	
0084	74 70	mov		a,#70h	;code for ‘p’
0086	12 00 6F	lcall		data_disp	;display it on LCD module
0089	74 48	mov		a,#48h	;code for ‘H’
008B	12 00 6F	lcall		data_disp	;display it on LCD module
008E	74 3D	mov		a,#3dh	;code for ‘=’
0090	12 00 6F	lcall		data_disp	;display it on LCD module
0093	22	ret			;return to main program
				display:	
0094	E9	mov		a,r1	;read digit after MSD ;of pH from r1
0095	60 05	jz		skip	
0097	44 30	orl		a,#30h	;convert it into ASCII

Program 36 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0099	12 00 6F	lcall		data_disp	;display it on LCD
009C	EA	skip: mov		a,r2	;read next digit
009D	44 30	orl		a,#30h	;convert it into ASCII
009F	12 00 6F	lcall		data_disp	;display it on LCD
00A2	74 2E	mov		a,#2eh	;load acc. with code for '.'
00A4	12 00 6F	lcall		data_disp	;display it on LCD
00A7					
00A7	EB	mov		a,r3	;read next digit
00A8	44 30	orl		a,#30h	;convert it into ASCII
00AA	12 00 6F	lcall		data_disp	;display it on LCD
00AD	EC	mov		a,r4	;read next digit
00AE	44 30	orl		a,#30h	;convert it into ASCII
00B0	12 00 6F	lcall		data_disp	;display it on LCD
00B3	22	ret			;return to main program ;delay subroutine
00B4				delay:	
00B4	7D 05	mov		r5,#05h	
00B6	7F FF	l8: mov		r7,#0ffh	
00B8	7E FF	l7: mov		r6,#0ffh	
00BA	DE FE	djnz		r6,$	
00BC	DF FA	djnz		r7,l7	
00BE	DD F6	djnz		r5,l8	
00C0	22	ret			
		end			;program ends

Program-37:

```
/* program for pH measurement using glass electrode*/

#include <stdio.h>
#include <reg51.h>

   sbit   x2 = P0^7;
   sbit   x3 = P0^6;
   sbit   x4 = P0^5;
   sbit   x5 = P0^4;

   sbit rs = P3^2;
   sbit   e = P3^3;

   int d1,d2,d3,d4,d5;
```

```
   void main (void)
   {
   void adc_7135(void);
   void lcd_comand(void);
   void lcd_data(void);
   void display(void);
   void lcd_int(void);
   int i;

   int v[] = {'p','H','='};
   lcd_int();
rep1:
         P1 = 0x02;          // bring cursor to the home position
    lcd_comand();

/* to display 'pH=' on LCD module */
    for (i = 1;i<3;i++)
    {
    P1 = v[i];
    lcd_data();
    }
    adc_7135();
        display();
        goto rep1;
    }

/* Sub-routine for A/D conversion */
   void adc_7135(void)
   {
do{
   d1 = P0;
   d1 = d1 & 0xf0;
   }while (d1!=0);

   d1 = P0 & 0x01;
   d1 = d1 | 0x30;

   while (x2!=1);
   d2 = P0 & 0x0f;
   d2 = d2 | 0x30;

while (x3!=1);
   d3 = P0 & 0x0f;
   d3 = d3 | 0x30;
```

```
   while (x4!=1);
   d4 = P0 & 0x0f;
   d4 = d4 | 0x30;

   while (x5!=1);
   d5 = P0 & 0x0f;
   d5 = d5 | 0x30;
   }
     /* Sub-routine to display pH of solution */
   int display()
   {
   if(d2 != 0x00);
   {
   P1 = d2;
   lcd_data();
         }
   P1 = d3;
   lcd_data();

       P1 = 0x2e;
   lcd_data();

   P1 = d4;
   lcd_data();

   P1 = d5;
   lcd_data();
   }
/* Sub-routine for initialization of LCD module */
   void lcd_int()
   {
   P1 = 0x38;
   lcd_comand();

   P1 = 0x0e;
   lcd_comand();

   P1 = 0x06;
   lcd_comand();

   P1 = 0x01;
   lcd_comand();
```

```
   P1 = 0x02;
   lcd_comand();
   }

/* sub-routine to send control words to the LCD module */

void lcd_comand(void)
   {
   int i;
   rs = 0;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
   /* sub-routine to send data to display on the LCD
       module*/

   void lcd_data(void)
   {
   int i;
   rs = 1;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
```

2

Microcontroller Based Conductivity Measurement System

AIM: To design and construct microcontroller based conductivity measurement system.

COMPONENTS: Microcontroller 89C51, A/D converter ICL7135, precision voltage regulator LM329, 2-5.1V Zener diodes, 100K pot, 0.1µF, 0.47µF, 1.0µF metalized polyester capacitors (each one), 1.0µF tantalum capacitor, 10µF capacitor, 100 KΩ multi-turn pot, 4–100KΩ, 8.2KΩ,100Ω, 27Ω resistors, 12.0 MHz crystal, switch, LCD module, bread board and connecting wires.

EQUIPMENT: Power supply +5V/3A.

INTRODUCTION: Electrolytic conductivity is a measure of the ability of a solution to carry an electrical current. Solution of electrolytes conducts an electrical current by the migration of ions under the influence of electric field. The conductance of a solution is the reciprocal of the electrical resistance. It is expressed in mhos(℧). Conductance C is directly proportional to the cross sectional area 'A' and inversely proportional to the length 'l' of a uniform conductor.

$$\text{Thus, } C \alpha A/l = K A/l$$

where, K is a proportionality constant called specific conductance or conductivity.' When A' and 'l' are numerically equal, the conductance becomes the specific conductance and hence it can be measured in terms of conductance.

PRINCIPLE: The operational amplifier in inverting configuration is used for measurement of conductivity of an electrolytic solution. It consists of an input resistance R_i and a feedback resistance R_f. Here the conductivity cell containing solution

acts as R_i. The output voltage of the op-amp is given by

$$V_o = -(R_f/R_i) * V_i$$
$$R_i = -(R_f/V_o) * V_i$$

When V_i=1V, and R_f=1000 Ohms

$$K = C = 1/R_i$$
$$K = V_o/1000$$
$$K = V_o \text{ m mhos}$$

So, the inverting amplifier is designed such that the reciprocal of the input resistance is equal to the output voltage, which is then acquired by the microcontroller through ADC and finally it is displayed on LCD module. Simultaneously the measurement of temperature of the solution is also incorporated in the system.

Hardware: The block diagram of the system is shown in Fig. 35. The functional blocks of the system are given below.

(a) Conductivity Cell
(b) Inverting Amplifier
(c) AC Source
(d) Precision Rectifier
(e) A/D Converter
(f) Temperature Measuring Circuit
(g) LCD display

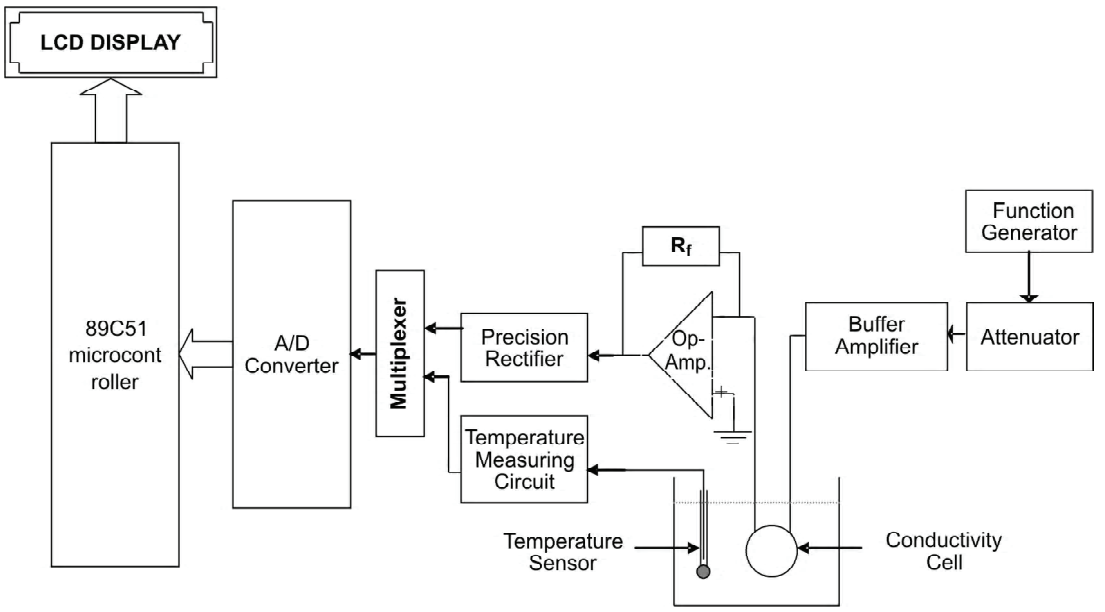


Fig. 35. Block diagram of microcontroller based conductivity measurement system.

The complete schematic of microcontroller based conductivity meter is shown in Fig. 36. The salient features of the individual blocks of the system are as follow.

(a) Conductivity Cell: The conductivity cell consists of a pair of electrodes that are firmly located in a constant geometry and which are immersed in an electrolytic solution whose conductivity is to be measured. The cell used in the present study consists of two platinum electrodes of $1cm^2$ cross sectional area that are separated by a distance of 1 cm. The cell constant of the cell used in the present study is 1.01.

(b) Inverting Amplifier: The inverting amplifier containing the conductivity cell as input resistance (R_i) is designed in such a way that the reciprocal of the input resistance is equal to the output voltage. Then the conductance of a solution becomes the output voltage of the operational amplifier in terms of milli-mhos(m℧). As the input applied to the conductivity cell is ac (1V/1kHz), the output of operational amplifier is also ac, and hence it cannot be directly applied to analog to digital converter. The signal is converted into dc by a precision rectifier and RC filter. The precision rectifier is constructed using two operational amplifiers A2 and A3 as shown in the schematic diagram.

(c) AC Source: AC source consists of a function generator, buffer and transformer. Function generator is designed using ICL8038 to generate sinusoidal signal for the excitation of the cell. Because of its excellent amplitude and frequency stability, it has been chosen in the present study. The amplitude and frequency of generator is exactly adjusted to 1V and 1kHz respectively. To avoid the loading effects, the sinusoidal signal from the function generator is applied to cell through an emitter follower.

(d) A/D Converter: The output of the inverting amplifier is given to the 4½-digit dual slope A/D converter ICL 7135 which is interfaced to the microcontroller. Figure 36 shows an interfacing between A/D converter and microcontroller. Port-0 of 89C51 is used to read the digit strobes and BCD data from A/D converter. The details of interfacing of A/D converter are explained in Experiment-13.

(e) Temperature Measuring Circuit: Accurate conductivity measurement depends on temperature compensation, because the conductivity varies with temperature. Hence, the temperature measurement is also incorporated in the system. An integrated circuit temperature sensor LM35 is used in the present study to measure temperature of the solution. The LM35 is Celsius sensor i.e., its output is ±10mV/°C. The output of the sensor is given to the 4½-digit A/D converter through an analog multiplexer.

(f) LCD Display: The system also contains the LCD module to display the conductivity and temperature of the solution. The interfacing of LCD module is given in Experiment-8.

System Working: The conductivity cell acts as input resistance (R_i) of the inverting amplifier, the output voltage of the amplifier represents the conductivity in milli-mhos of the solution under investigation. Since input of the inverting amplifier is 1V/1kHz sinusoidal signal, the output of the inverting amplifier is also a sinusoidal signal. The precision rectifier and RC filter are used to convert sinusoidal output voltage of the inverting amplifier into dc voltage. Then it is given to the A/D converter though an analog multiplexer, which helps in selecting one of the two

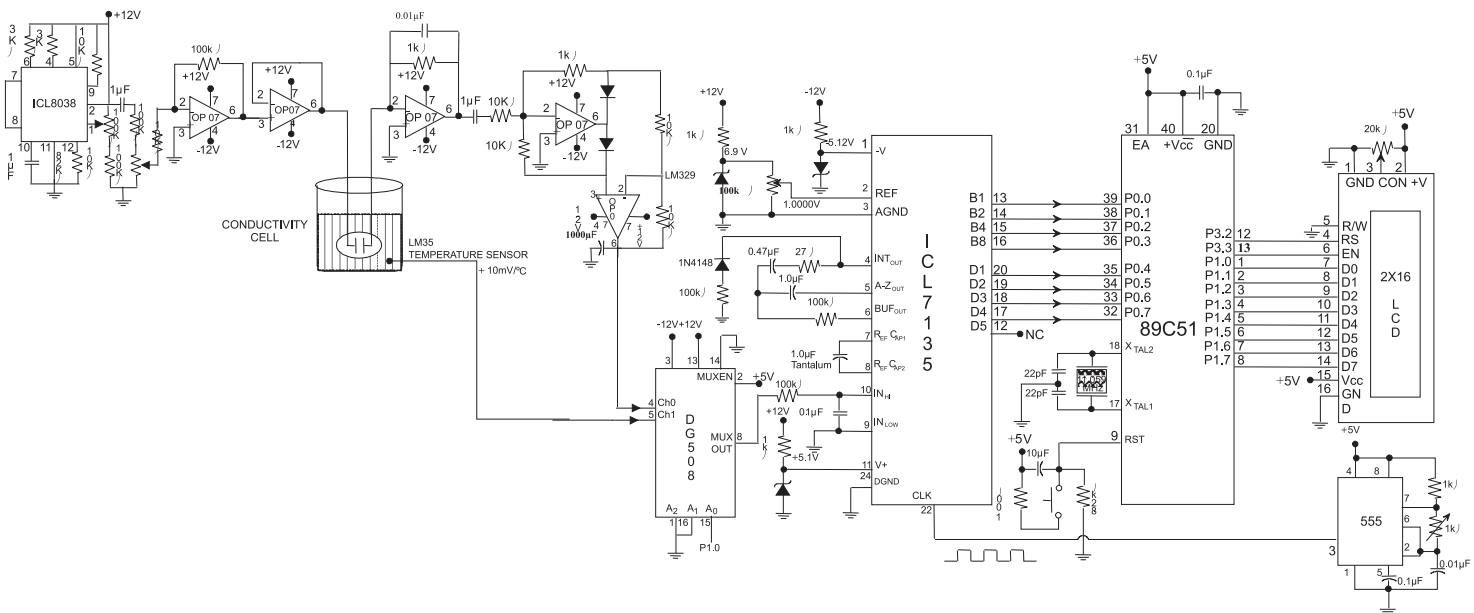


Fig. 36. Circuit Schematic of microcontroller based conductivity measurement system.

parameters i.e., conductivity and temperature. The A/D converter provides the digital data corresponding to the parameter under measurement. The same is read by the microcontroller through port-0 of microcontroller 89C51. The port bit P1.0 of microcontroller is used to select one of the two measuring parameters. The microcontroller sends HIGH on P1.0 to select the conductivity input, and then acquires it through A/D converter and displays the same on the liquid crystal display. Then the microcontroller selects the temperature by sending LOW on P1.0, and then it acquires and displays the same on liquid crystal display. The above process is repeated continuously.

Software: The role of the software in the present study is given below.

(a) To initialize the liquid crystal display.
(b) Select the conductivity through the relay and read it through digital data through A/D converter process it and displays the same on the liquid crystal display.
(c) Select the temperature through relay and read it through A/D converter and display on LCD module.
(d) Repeat the above procedure from step (b).

The software is written in the microcontroller embedded 'C' language. The flow chart of the program (Program-38) is presented in the Fig. 37.

Program 38 : Program for measurement of pH of a solution

Address	Opcode	Label	Mnemonic	Operand	Comments
		org	0000h		
					;Initialization of LCD display
0000	75 81 40	mov		sp,#40h	;Initialize stack pointer ;with the starting address 40h
0003	74 38	mov		a,#38h	;function set control word
0005	12 00 5A	lcall		enable	;send to LCD module and
0008	74 0E	mov		a,#0eh	;display ON/OFF control word
000A	12 00 5A	lcall		enable	
000D	74 06	mov		a,#06h	;entry mode set control word
000F	12 00 5A	lcall		enable	
0012	74 01	mov		a,#01h	;clear display control word
0014	12 00 5A	lcall		enable	
0017	74 02	mov		a,#02h	;return home control word
0019	12 00 5A	lcall		enable	
001C	12 00 84	lcall		pH_disp	;call subroutine to display ;'pH=' on LCD module
001F	12 00 2D	lcall		adc	;call subroutine A/D conversion
0022	12 00 94	lcall		display	;call subroutine to display

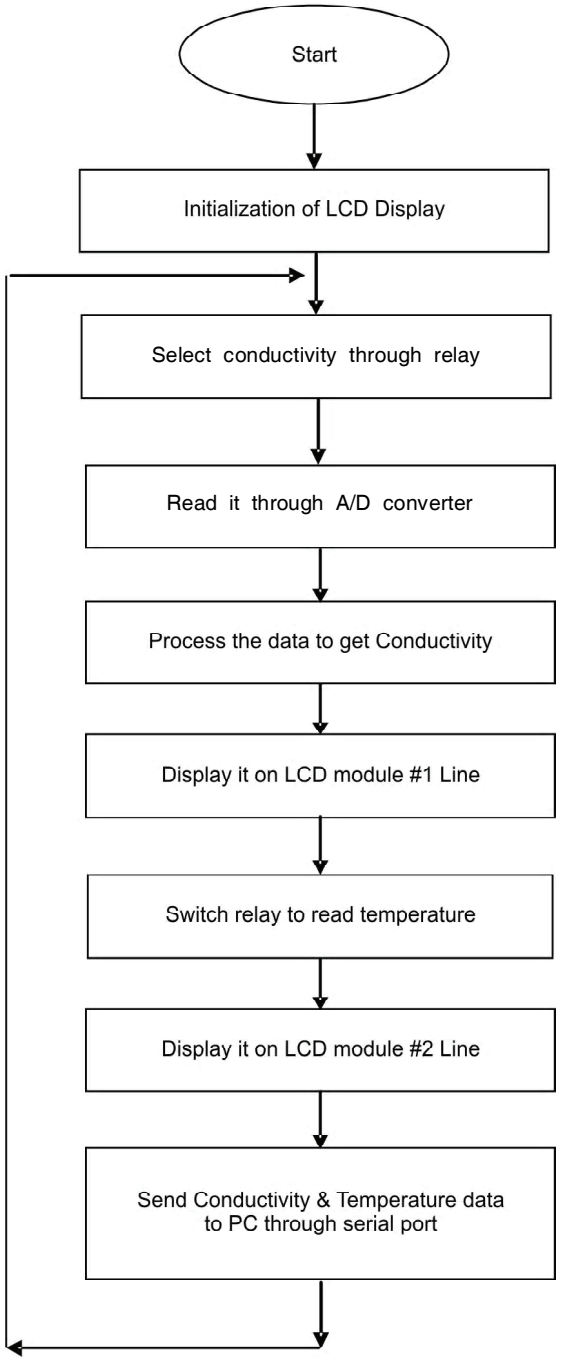

Start
Initialization of LCD Display
Select conductivity through relay
Read it through A/D converter
Process the data to get Conductivity
Display it on LCD module #1 Line
Switch relay to read temperature
Display it on LCD module #2 Line
Send Conductivity & Temperature data
to PC through serial port

Fig. 37. Flow chart for measuring and displaying conductivity and temperature of a solution.

Program 38 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
					;to display pH
0025	74 02		mov	a,#02h	;bring cursor to left
0027	12 00 5A		lcall	enable	;corner of the display
002A	02 00 1C		ljmp	l1	;repeat the program ;program for ICL7135 A/D converter
002D				adc:	
002D	E5 80	x0:	mov	a,p0	;read digit strobes and ;data from A/D converter
002F	54 F0		anl	a,#0f0h	;mask data
0031	B4 00 F9		cjne	a,#00h,x0	;check whether D1-D4 digit ;strobes are LOW, if not ;go to location x0
0034	E5 80		mov	a,p0	;if all digit strobes are ;LOW, the read data from ;A/D converter
0036	54 0F		anl	a,#0fh	;mask digit strobes
0038	F8		mov	r0,a	;store MSD data in r0
0039	30 84 FD		jnb	p0.4,$	;poll digit strobe D4 until ;it goes HIGH
003C	E5 80		mov	a,p0	;read digital data
003E	54 F0		anl	a,#0f0h	;mask digit strobes
0040	F9		mov	r1,a	;store data in r1
0041	30 85 FD		jnb	p0.5,$	;poll digit strobe D3 until ;it goes HIGH
0044	E5 80		mov	a,p0	;read digital data
0046	54 0F		anl	a,#0fh	;mask digit strobes
0048	FA		mov	r2,a	;store data in r2
0049	30 86 FD		jnb	p0.6,$	;poll digit strobe D2 until ;it goes HIGH
004C	E5 80		mov	a,p0	;read digital data
004E	54 0F		anl	a,#0fh	;mask digit strobes
0050	FB		mov	r3,a	;store data in r3
0051	30 87 FD		jnb	p0.7,$	;poll digit strobe D1 until ;it goes HIGH
0054	E5 80		mov	a,p0	;read digital data
0056	54 0F		anl	a,#0fh	;mask digit strobes
0058	FC		mov	r4,a	;store data in r4
0059	22		ret		;return to main program
				enable:	
005A	F5 90		mov	p1,a	;send control word to display
005C	C2 B4		clr	p3.2	;RS=0
005E	D2 B5		setb	p3.3	;enable=1
0060	75 F0 FF		mov	b,#0ffh	;small delay
0063	D5 F0 FD		djnz	b,$	
0066	C2 B5		clr	p3.3	;enable=0
0068	75 F0 FF		mov	b,#0ffh	;small delay

Program 38 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
006B	D5 F0 FD	djnz		b,$	
006E	22	ret			;return to main program
				data_disp:	
006F	F5 90	mov		p1,a	;send character code to display
0071	D2 B4	setb		p3.2	;RS=1
0073	D2 B5	setb		p3.3	;enable=1
0075	75 F0 FF	mov		b,#0ffh	;small delay
0078	D5 F0 FD	djnz		b,$	
007B	C2 B5	clr		p3.3	;enable=0
007D	75 F0 FF	mov		b,#0ffh	;small delay
0080	D5 F0 FD	djnz		b,$	
0083	22	ret			;return to main program
				pH_disp:	
0084	74 70	mov		a,#70h	;code for 'p'
0086	12 00 6F	lcall		data_disp	;display it on LCD module
0089	74 48	mov		a,#48h	;code for 'H'
008B	12 00 6F	lcall		data_disp	;display it on LCD module
008E	74 3D	mov		a,#3dh	;code for '='
0090	12 00 6F	lcall		data_disp	;display it on LCD module
0093	22	ret			;return to main program
				display:	
0094	E9	mov		a,r1	;read digit after MSD
					;of pH from r1
0095	60 05	jz		skip	
0097	44 30	orl		a,#30h	;convert it into ASCII
0099	12 00 6F	lcall		data_disp	;display it on LCD
009C	EA	skip:	mov	a,r2	;read next digit
009D	44 30	orl		a,#30h	;convert it into ASCII
009F	12 00 6F	lcall		data_disp	;display it on LCD
00A2	74 2E	mov		a,#2eh	;load acc. with code for '.'
00A4	12 00 6F	lcall		data_disp	;display it on LCD
00A7					
00A7	EB	mov		a,r3	;read next digit
00A8	44 30	orl		a,#30h	;convert it into ASCII
00AA	12 00 6F	lcall		data_disp	;display it on LCD
00AD	EC	mov		a,r4	;read next digit
00AE	44 30	orl		a,#30h	;convert it into ASCII
00B0	12 00 6F	lcall		data_disp	;display it on LCD
00B3	22	ret			;return to main program
					;delay subroutine
00B4				delay:	
00B4	7D 05	mov		r5,#05h	
00B6	7F FF	l8:	mov	r7,#0ffh	
00B8	7E FF	l7:	mov	r6,#0ffh	
00BA	DE FE	djnz		r6,$	
00BC	DF FA	djnz		r7,l7	
00BE	DD F6	djnz		r5,l8	
00C0	22	ret			
		end			;program ends

Program-39:

```
/* program for pH measurement using glass electrode*/

#include <stdio.h>
#include <reg51.h>

   sbit   x2 = P0^7;
   sbit   x3 = P0^6;
   sbit   x4 = P0^5;
   sbit   x5 = P0^4;

   sbit rs = P3^2;
   sbit  e = P3^3;

   int d1,d2,d3,d4,d5;

   void main (void)
   {
   void adc_7135(void);
   void lcd_comand(void);
   void lcd_data(void);
   void display(void);
   void lcd_int(void);
   int i;

   int v[] = {'p','H','='};
   lcd_int();
rep1:
        P1 = 0x02;          // bring cursor to the home position
    lcd_comand();

/* to display 'pH=' on LCD module */
    for (i = 1;i<3;i++)
    {
    P1 = v[i];
    lcd_data();
    }
    adc_7135();
        display();
        goto rep1;
    }
```

```
/* Sub-routine for A/D conversion */
   void adc_7135(void)
   {
do{
   d1 = P0;
   d1 = d1 & 0xf0;
   }while (d1!=0);

   d1 = P0 & 0x01;
   d1 = d1 | 0x30;

   while (x2!=1);
   d2 = P0 & 0x0f;
   d2 = d2 | 0x30;

while (x3!=1);
   d3 = P0 & 0x0f;
   d3 = d3 | 0x30;

   while (x4!=1);
   d4 = P0 & 0x0f;
   d4 = d4 | 0x30;

   while (x5!=1);
   d5 = P0 & 0x0f;
   d5 = d5 | 0x30;
   }
     /* Sub-routine to display pH of solution */
   int display()
   {
   if(d2 != 0x00);
   {
   P1 = d2;
   lcd_data();
         }
   P1 = d3;
   lcd_data();

       P1 = 0x2e;
   lcd_data();

   P1 = d4;
   lcd_data();
```

```
   P1 = d5;
   lcd_data();
   }
/* Sub-routine for initialization of LCD module */
   void lcd_int()
   {
   P1 = 0x38;
   lcd_comand();

   P1 = 0x0e;
   lcd_comand();

   P1 = 0x06;
   lcd_comand();

   P1 = 0x01;
   lcd_comand();

   P1 = 0x02;
   lcd_comand();
   }

/* sub-routine to send control words to the LCD module */

void lcd_comand(void)
   {
   int i;
   rs = 0;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
   /* sub-routine to send data to display on the LCD
       module*/
   void lcd_data(void)
   {
   int i;
   rs = 1;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
```

3

Microcontroller Based Liquid Level Measurement System

AIM: To design and construct microcontroller based liquid level measurement system.

COMPONENTS: Capacitance type liquid level transducer, 89C51 microcontroller, monostable multivibrator-74LS121, 100 kΩ multi-turn pot, 12.0 MHz crystal, switch, LCD module, bread board and connecting wires.

EQUIPMENT: Power supply +5V/3A.

PRINCIPLE: The circuit schematic of microcontroller based liquid level measurement system is shown in Fig. 38. A capacitive type transducer having a metallic tank with two co-axial electrically conducting electrode tubes which are acting as plates of the capacitor and the liquid (water) whose level is to be measured serves as a dielectric medium in a cylindrical capacitor. The changes in the capacitance are linear with the changes in the liquid level. The change in the capacitance of the transducer is converted into change in ON-time by connecting transducer as C in the monostable multivibrator circuit. The microcontroller measures the ON-time of output of the monostable multivibrator, and then it calculates liquid level and displays it on LCD module.

Hardware: The liquid level measurement system designed for the present study consists of the following elements.

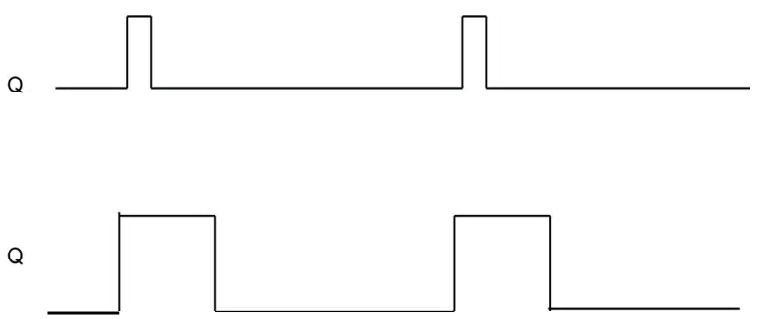
a. Liquid level transducer
b. Monostable multivibrator
c. Microcontroller 89C51
d. LCD module

(a) Liquid Level Transducer: A capacitive transducer has a cylindrical metallic tank with base and it comprises of two vertically coaxial, electrically conductive tubes constitute electrodes of the capacitor are immersed in the liquid filled tank. The liquid acts as a dielectric medium. In this transducer the area and the distance between the two electrodes are kept constant, only the liquid i.e. dielectric medium is variable. As the level of the liquid between the electrodes increases, capacitance increases, because the dielectric constant is directly proportional to the capacitance, similarly if the liquid level is decreases, the capacitance between two electrodes also decreases. The capacitance is a function of the height of the liquid. Thus the capacitance becomes a linear function of the liquid level. The capacitance of the tube is measured through the probes.

(b) Signal Conditioner: In the present study the capacitance of the transducer acts as ON-time determining capacitor of a monostatble multivibrator as shown Fig. 38. A monostable multivibrator is a device that outputs a pulse of predetermined length. Every time the input signal level changes from a logical 0 to a logical 1. The length of the output pulse is usually determined by a resistor–capacitor network. The ON-time of the output of monostable multivibrator is given by the equation,

$$T = 0.7\ RC$$

Here, 'R' is fixed and the 'C' varies as the liquid level in the tank varies and hence, the ON-time of the output of the monostable multivibrator. The following figures show the output of monostable multivibrator for different liquid levels. The monostable multivibrator is designed using 74LS121.



(c) Microcontroller: Here, 89C51 microcontroller has been employed to measure the liquid level. The role assigned to the microcontroller in the present study is to acquire and measure the On-time of output of the monostatble multivibrator and calculate the liquid level using the following equations:

$$T = 0.7RC$$

$$C = \frac{T}{0.7R}$$

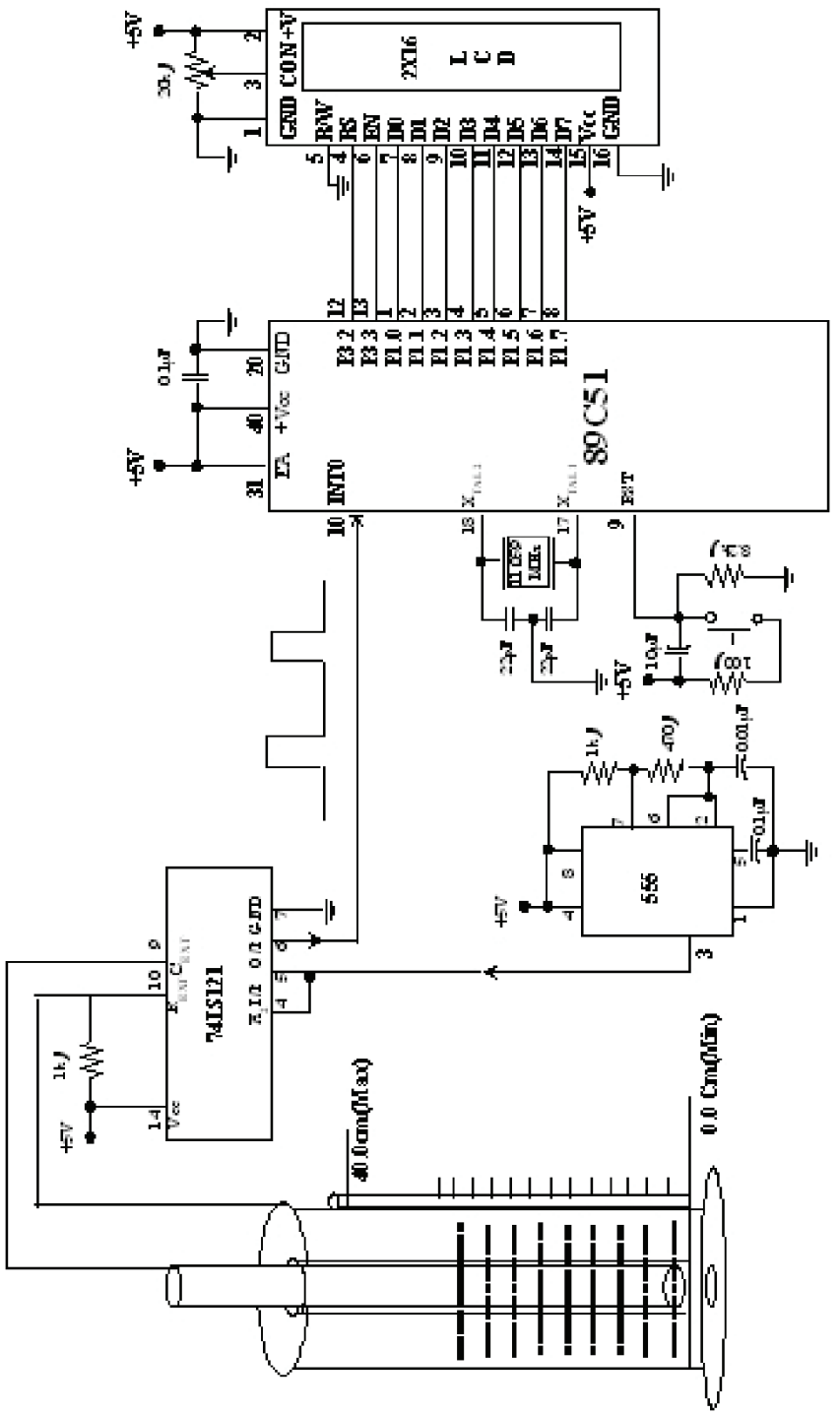


Fig. 38. Microcontroller based liquid level measurement system.

then microcontroller calculates the liquid level by substituting C in the following equation

$$Y = mX + C$$

which is derived from the capacitance verses liquid level plot using least square curve fitting method.

where,	X	is liquid level
	Y	is capacitance
	m	slope
	C	intercept

and the processed data is displayed on LCD module, which is the level of the liquid in the tank.

Working of the System: In the microcontroller based liquid level measuring system, the transducer acts as capacitor. The liquid whose level is to be measured acts as dielectric. The output of the transducer is in the order of nano-Farads and it is signal conditioned by using the monostable multivibrator. The ON-time of the pulse of monostable multivibrator is directly proportional to the capacitance and the capacitance is again directly proportional to the level of the liquid in the tank. ON-time of the pulse is measured using the microcontroller. Microcontroller uses its two timers to measure the ON time of the output of monostable multivibrator. From the ON-time (T), the microcontroller calculates capacitance by the relation, C=T/(0.7R) where R is a fixed value, i.e. R = 50.142kΩ. The C is substituted in the following equation to calculate the liquid level;

$$Y = mX+C$$

or

$$X = (Y-C) / m$$

where X-The liquid level, Y- Capacitance, m = (slope) and C = (intercept)

The slope and intercept are derived from the least square curve fitting method by measuring capacitance of the transducer for liquid height from 1 cm to 100 cm. After measuring the liquid level, the microcontroller will display it on the LCD module in terms of cm. The accuracy of the system is ± 0.1cm.

Software: The system software is developed in the microcontroller 89C51 embedded 'C' language is given Program-40 and is divided into five parts.

1. Program for the measurement of ON-time of the monostable multivibrator.
2. Program for calculating the capacitance.
3. Program for calculating the liquid level.
4. Program for displaying the liquid level on LCD module.
5. Program for sending the liquid level to the computer through the serial port.

The working of the ON-time measurement program is described in experiment 17. An external control/gate input signal INT0 is used to measure the ON-time of the output of monostatble multivibrator. The measured ON-time of the monostable multivibrator is substituted in the equation C = T/(0.7R) to calculate capacitance and C is substituted in the equation

$$X = (Y- C)/m$$

Finally the liquid level will be displayed on the LCD module.
Results: The measurement accuracy is ±1mm. The unit is highly reliable, very compact, and versatile and is operated by single power supply of +5V. The liquid level measurement system's readings are tabulated in the Table 21.

Table 21

Sr. No.	Level indicated on Scale in cm	Level measured in cm
1	0.0	0.0
2	10.0	10.1
3	20.0	20.2
4	30.0	30.0
5	40.0	40.1

Program 40:

```
/* program for liquid level measurement using INT0 interrupt and
timer0 */
#include <stdio.h>
#include <reg51.h>

   sbit   rs = P3^2;
   sbit    e = P3^3;

   int d,d1,d2,d3,d4,d5;
   float v;

   void main (void)
   {
        void lcd_int(void);
   void lcd_comand(void);
   void lcd_data(void);
   void data_conv(void);
   void display(void);
   int i, hc,lc,T,C,levl;
```

```
   int c[] = {'L','e','v','e','l','='};
   lcd_int();           //initialization LCD module
   flag = 0;                  //clear user flag
   TMOD = 0x09;               //timer-0 16-bit timer
   TH0 = 0x00;                //clear timer-0
   TL0 = 0x00;
   TCON = 0x11;               //start timer-0
                              //external-0 interrupt is
                              //transition activated
   IE = 0x81                  //enable external-0 interrupt
                              //enable global interrupt enable bit
   while(1)
   {
   while(flag == 0);          //stay in loop until F0 goes high
   flag = 0;                  //Clear F0 flag

   lc = TL0;                  //read low byte timer-0
   hc = TH0;                  //read high byte of timer-0
   T = hc*256+lc;             //assign 16-bit number to 'T'
   C = T/(0.7R);              //calculate 'C'
   Levl = (C- 1.02)/.321      //calculate liquid level using relation
                              //between capacitance and liquid level
v = levl;
data_conv();                  //unpack 16-bit number

   P1 = 0x02;                 //bring cursor to left corner of display
   lcd_comand();              //send to display

   for (i = 1;i<7;i++)        //display 'Level=' on LCD module
   {
   P1 = c[i];
   lcd_data();
   }
        display();            //display value of level

   P1 = 'C';                  //display 'Cm'
   lcd_data();
P1 = 'm';
   lcd_data();
        while(INT0 == 0);     //start measurement at low to high
        while(INT0 != 0);     //transition of the input signal
   TMOD = 0x09;
   TH0 = 0x00;
   TL0 = 0x00;
```

```
   TCON = 0x11;
   IE   = 0x81;
   }
   }
    void p_width_isr(void) interrupt 0
   {
   IE = 0x00;    //disable all interrupts
   TCON = 0x00;  //stop timer-0
   flag = 1;     //set user flag
   }
 /* subroutine to separate digits */

   void data_conv(void)
   {
   v = v*10000.0;
   d = v;
   d1 = d/10000;

   d2 = d % 10000;
   d2 = d2/1000;

   d3 = d % 1000;
   d3 = d3/100;

   d4 = d % 100;
   d4 = d4/10;

   d5 = d % 100;
   d5 = d5/10;
   }
/* subroutine to display liquid level on LCD module */
   void display(void)
   {
   d1 = d1+0x30;
   P1 = d1;
   lcd_data();

   d2 = d2+0x30;
   P1 = d2;
   lcd_data();

P1 = 0x2e;
   lcd_data();
```

```
   d3 = d3+0x30;
   P1 = d3;
   lcd_data();

   d4 = d4+0x30;
   P1 = d4;
   lcd_data();

   d5 = d5+0x30;
   P1 = d5;
   lcd_data()
       }

   /* LCD initialization subroutine */
void lcd_int()
   {
   P1 = 0x38;
   lcd_comand();

   P1 = 0x0e;
   lcd_comand();

   P1 = 0x06;
   lcd_comand();

   P1 = 0x01;
   lcd_comand();

   P1 = 0x02;
   lcd_comand();
   }
  /* subroutine for sending control words to LCD module */
   void lcd_comand()
   {
   int i;
   rs = 0;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
   /* subroutine for sending data to display on LCD module */
   void lcd_data()
   {
```

```
int i;
rs = 1;
e =  1;
for (i = 1;i<1000;i++);
e = 0;
for (i = 1;i<1000;i++);
}
```

4

Microcontroller Board

AIM: To design a microcontroller board for measurement and control applications.

COMPONENTS: Microcontroller 8031, 8K-EPROM 2764, 8K-RAM 6264, programmable peripheral interface 8255, latch 74HC573, decoder 74LS138, serial line converters MC1488, MC1489, 10-0.1μF and berg connectors.

EQUIPMENT: Power supply +5V/3A

Hardware: The circuit schematic of the microcontroller board is shown in the Fig. 39. It contains all the features required for it to function as stand-alone system. The board contains the following features.

- Microcontroller (8031)
- 8K bytes of program memory (2764)
- 8K bytes of program/data memory (6264)
- Programmable peripheral interface (8255) and
- Serial port and
- LCD module

Interfacing of Peripherals: The multiplexed lower order address/data bus of microcontroller is separated into address and data bus using latch 74HC573 and ALE signal. In the first part of the machine cycle the microcontroller sends address on port-0 and logic HIGH on ALE pin. The HIGH on ALE enables latch and the address will be transferred to the output lines of the latch. In the later part of the machine cycle the ALE goes LOW, then port-0 acts as data bus of the microcontroller. The lines (D_0-D_7) of microcontroller are connected to the data lines of memory chips i.e. EPROM and RAM and 8255. Since both EPROM and RAM require 13 address lines to address 8K bytes, the address lines A_0 to A_{12} of microcontroller are connected to the address lines of EPROM and RAM. The remaining address lines A_{13} to A_{15} are used for selecting the chips with the help of a 3 to 8 decoder 74LS138.

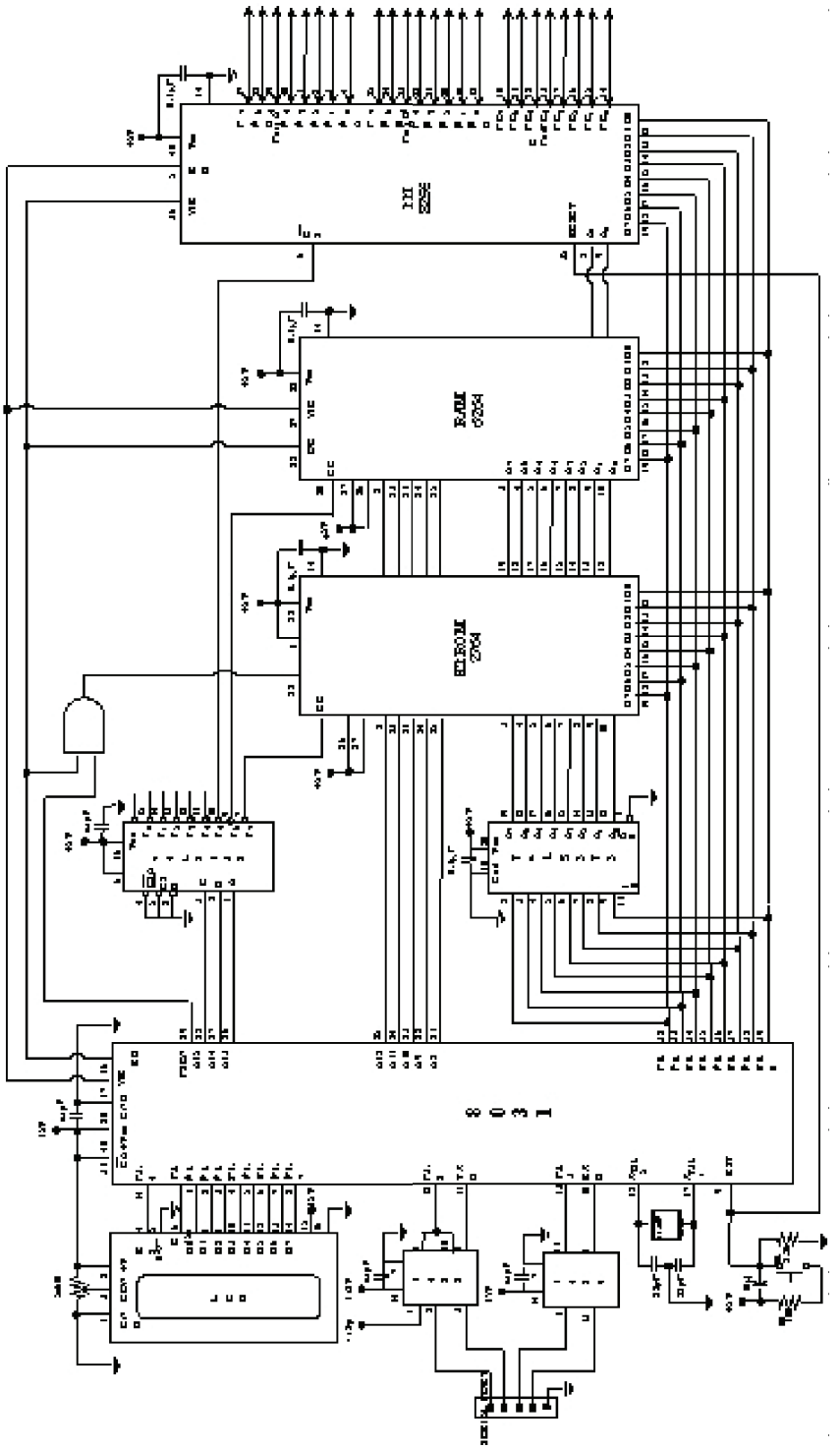


Fig. 39. Circuit schematic of the microcontroller board.

The RD and PSEN of the microcontroller are combined with the help of an AND gate (74LS08), the output of the AND gate is connected to the output enable (OE) of EPROM, this arrangement enables to read data from the EPROM. Sending the address to EPROM can access the contents of the memory at any location. Since, the read and write operations will take place in the RAM, the RD and WR signals are connected to the OE and WE pins of the chip respectively. Y0 and Y1 of the decoder outputs are connected to CS of EPROM and RAM respectively. The address map of EPROM and RAM are given below.

Program Memory	2764 (8K)	0000H - 1FFFH
Data Memory	6264 (8K)	2000H - 3FFFH

Since there are three ports in 8255, two address lines are sufficient to select all the ports. The address lines A_0, A_1 of the microcontroller are connected to the A_0, A_1 of 8255, which provide port selection. The RD, WR and RESET pins of the microcontroller are connected to RD, WR and of 8255 to synchronize its read/write operations with the microcontroller. The data lines D_0 to D_7 of microcontroller are connected to 8 data lines of 8255 for transmitting or receiving the data from the microcontroller. The CS of the 8255 is connected to the Y2 of the decoder (74LS138). All the three port lines are brought to 26 pin berg connector. The addresses of ports of 8255 are given below.

Port-A	4000H
Port-B	4001H
Port-C	4002H
C.R.	4003H

Serial Port: The ICs MC1488 and MC1489 are incorporated in the microcontroller board for serial communication with the host computer. The TXD and RXD lines of microcontroller are connected to the corresponding lines of the ICs MC1488 and MC1489. The ICs will increase the digital logic levels from 0 to 5V to serial communication standard levels +12 to –12V levels. The P3.3 and P3.4 of microcontroller are used for handshaking signals for serial communication. The serial port of the microcontroller enables the host computer to read the parameter and store it in the specified file for further analysis. This facility makes the microcontroller based instrument suitable for research applications.

The board can be tested by connecting binary counter LEDs display to port-A of 8255 as shown Fig. 11. The program for binary counter is given in Program-41.

Program 41: Program for binary counter

Address	Opcode	Label	Mnemonic	Operand	Comments
		org	0000h		;starting of the program
0000	90 40 03	mov		dptr,#4003h	;load DPTR with 8255 control ;register address
0003	74 80	mov		a,#80h	;load acc. with control word ;to make all ports as output ;ports
0005	F0	movx		@dptr,a	;send control word into the ;control register
0006	90 40 00	mov		dptr,#4000h	;load DPTR with port-A address
0009	E4	clr		a	;clear accumulator
000A	F0	movx		@dptr,a	;send acc. contents to port-A
000B	12 00 12	dis: lcall		delay	;call delay subroutine
000E	04	inc		a	;increment acc.
000F	02 00 0B	ljmp		dis	;jump to label 'dis' to send the ;contents of acc. to port-A ;delay subroutine
0012	78 08	delay: mov		r0,#08h	
0014	79 FF	l3: mov		r1,#0ffh	
0016	7A FF	l2: mov		r2,#0ffh	
0018	DA FE	l1: djnz		r2,l1	
001A	D9 FA	djnz		r1,l2	
001C	D8 F6	djnz		r0,l3	
001E	2	ret			;return to main program

5

Microcontroller Based DC Motor Speed Control System

AIM: To design and construct microcontroller based DC motor speed control system.

COMPONENTS: 89C51, A/D converter- ADC0809, transducer (tachogenerator), precision voltage regulator LM329, LM308, MAX232, NE555 timer, 2-1kΩ resistors, 2-10kΩ pots, 10µF, 0.1µF, 0.01µF, 12.0 MHz crystal, switch, LCD module, bread board and connecting wires.

EQUIPMENT: Personal computer, CRO.

PRINCIPLE: The block diagram of microcontroller based DC motor speed control system is presented in Fig. 40. The rotational speed of the DC motor is converted into series of pulses using tachogenerator. Frequency of these pulses is directly proportional to the speed of the DC motor. These pulses are converted into voltage using frequency to voltage (F/V) converter; the output of the F/V converter is directly proportional to the speed of the DC motor. This analog voltage is converted into digital data using analog to digital (A/D) converter. The output of the A/D converter is directly proportional to the speed of the DC motor. The microcontroller acquires the digital data from A/D converter and converts it into actual speed of the DC motor by using least square curve fitting method. The equation to convert A/D output into speed in rpm is given below.

$$y = mx+c$$

where,

	x	=	digital data from A/D converter
	m	=	slope of frequency v/s voltage graph
	c	=	intercept on Y-axis
	y	=	speed of DC motor in RPM

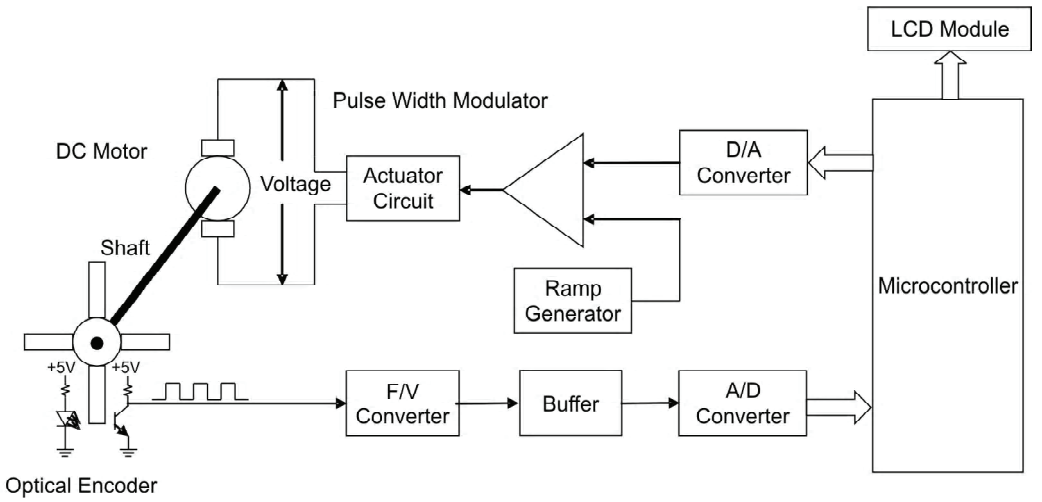


Fig. 40. The block diagram of microcontroller based DC motor speed control system.

The measured speed is displayed on the LCD module. The above equation is fitted by taking A/D converter output for different rotational speeds of DC motor. To control the speed of a DC motor, the measured speed is subtracted from the desired speed to obtain error. This error is applied to PID program, which solves the equation and produces a digital data, which is given to D/A converter. The output of the D/A converter is applied to non-inverting input of comparator whose another input is connected to a ramp signal. This comparator acts as a pulse width modulator (PWM) and who's ON time is proportional to D/A converter output. The output of PWM controls the power to the DC motor. This process is repeated continuously till the DC motor attains the desired speed.

Hardware: Figure 41 shows the complete circuit schematic of microcontroller based DC motor speed control system which consists of the following elements.

a. DC motor
b. Tachogenerator
c. Frequency to voltage converter
d. Analog to digital converter
e. Microcontroller
f. LCD module
g. Digital to analog converter and pulse width modulator
h. Actuator
i. Serial port

a. DC motor

The DC motor with the following specifications is used in the present study;

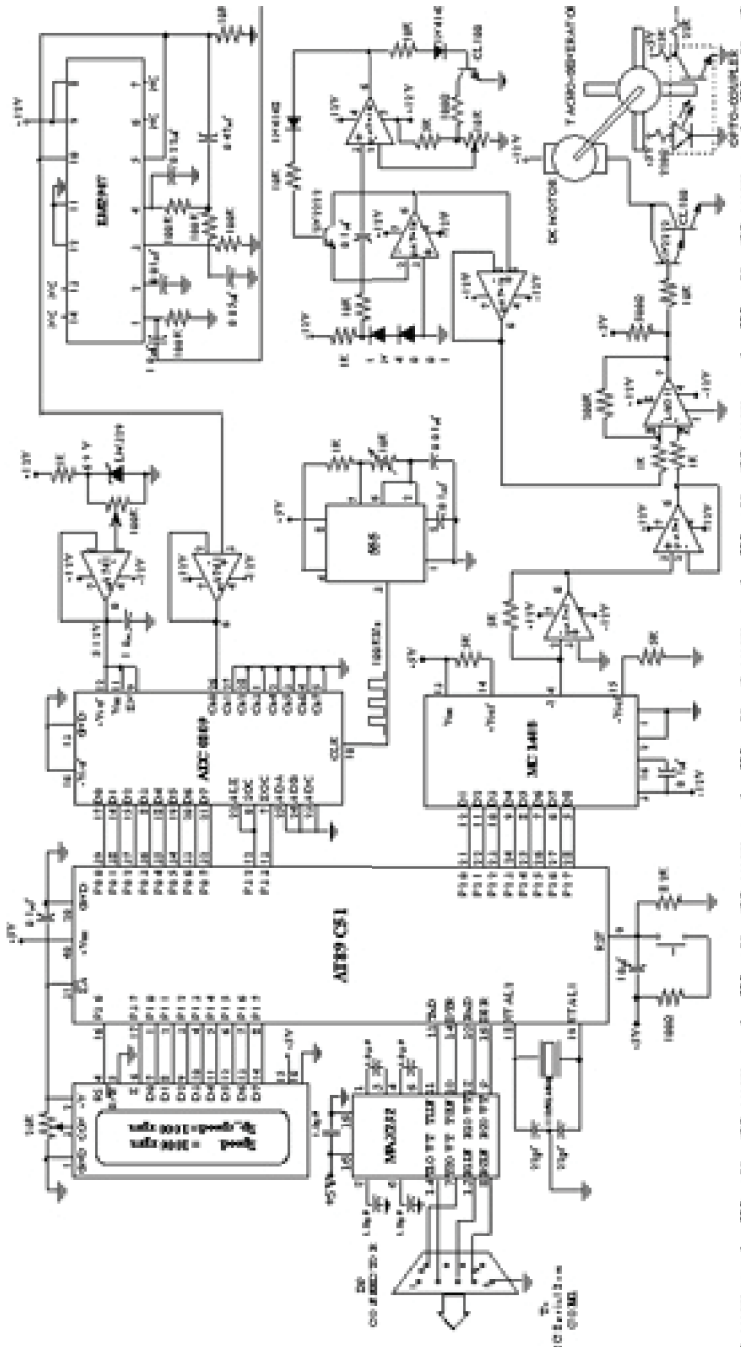


Fig. 41. Circuit schematic of microcontroller based DC motor speed control system.

Rated voltage	12 Volt DC
Rated current	200mA at no load
	290mA at full load
Torque	50 gm-cm
Max speed	2400 rpm

b. Tachogenerator

The shaft of the DC motor is connected to the tachogenerator. The tachogenerator is a transducer, which generates a series of pulses whose frequency is proportional to the speed of DC motor. The tachogenerator consists of a LED, phototransistor and a slotted aluminum disk connected to the motor shaft. Whenever light from the LED falls on the base of the phototransistor then it starts conducting and a low pulse is obtained at the collector of phototransistor, and when no light falls on the base of the phototransistor a high pulse is obtained. In this way, series of pulses is generated for continuous rotation of the aluminum disk.

c. Frequency to voltage converter

An AC signal proportional to the shaft speed is obtained from a tachogenerator, which is in the form of series of pulses. This series of pulses is converted into voltage by using F/V converter constructed using LM 2907. The output of F/V is a DC voltage which is directly proportional to the frequency of the input signal.

d. Analog to digital converter

The analog voltage from the output of F/V converter is applied to A/D converter, which converts analog voltage into digital data. This digital data is proportional to the speed of the DC motor. The microcontroller acquires this digital data through port-0. Figure 41 shows the interfacing of A/D converter ADC0809 with microcontroller. The buffered output of F/V converter is connected to channel0 (Ch0) of A/D converter. Channel select inputs (ADA, ADB, and ADC) are tied low to select Ch0 permanently. The data lines D0 to D7 of A/D converter are connected to port P0.0 to P0.7 of microcontroller. The conversion is initiated by sending a HIGH to LOW transition pulse to ALE and SOC which are connected together to port P3.4 of microcontroller. The EOC of A/D converter indicates whether conversion is completed or not. If EOC is LOW then conversion is going on else conversion is completed and digital data will be available on D0 to D7. The status of EOC is polled by P3.5 pin of microcontroller.

e. Microcontroller

An 8-bit microcontroller (AT89C51) has been used. It makes the system versatile and economical because of availability of on-chip program and data memory and I/O ports. The role assigned to the microcontroller in the present study is to acquire the digital data from A/D converter, calculate speed of the DC motor and display it on LCD module, evaluate the error by subtracting measured speed from a set-point speed, solve the PID control algorithms. To enable the microcontroller to implement the PID control algorithms, the continuous differential equations are to be converted to discrete difference equations as given below. The velocity algorithms are given by the equations

$$V_n = V_{n-1} + K_p(e_n - e_{n-1}) + K_i\, e_n\, T + K_d/T\, [(e_n - 2e_{n-1} + e_{n-2})] \quad \ldots\ldots\ (v)$$

The above equation (V_n) is a standard PID difference equation. At any instant of time the current value of the PID output V_n is calculated based on the previous value of the PID output V_{n-1}, current error e_n, previous error e_{n-1}, previous to the previous error e_{n-2}, the cycle time T and weighing constants(K_p, K_i, K_d). The output of control program is a digital value, which is applied to the D/A converter.

f. LCD module

A 2-line, 16 characters LCD module is used to display the measured speed. Figure 43 presents the interfacing of display module with the microcontroller. The data bus D0-D7 of display module is connected to port P1.0-P1.7 and control lines RS and E are connected to the port P3.4 and P3.5 respectively. The command word is sent to command register by making RS LOW and data can be sent to display RAM by making RS HIGH. A HIGH pulse of about 40μsec duration is sent to the pin E of the display to fetch/send the data/command word to/from the display. Another control line R/W of the display is connected to ground and third pin is connected to the wiper of the potentiometer to vary the contrast of the display.

g. Digital to analog converter and pulse width modulator

The D/A converter MC1408 is interfaced with the microcontroller through port-2. The output of D/A converter is connected to the non-inverting terminal of comparator (LM311) and inverting input of comparator is connected to the output of ramp generator circuit which is designed using three op-amps (741a-c) as shown in Fig. 41. The 'ON' time of the comparator depends on the output of the D/A converter. If the output of the D/A converter is large, then 'ON' time of the comparator output will be more, similarly if the output of the D/A converter is small, then 'ON' time of the comparator output will be less.

h. Actuator

The final power control element is an actuator. In this work the actuator used is a Darlington pair transistor, which acts as an ON-OFF electronic switch for DC motor. It also works as a buffer between PWM and DC motor. If a single transistor is used, the maximum current obtained through it will be of the order of 50mA, which is not sufficient to drive a high power device such as DC motor. For this reason Darlington pair transistors are used to drive the DC motor.

i. Serial port

The MAX232 is also incorporated in the microcontroller board for serial communication with the host computer. This enables the user to tune PID parameters and change the set-point. The computer will read the speed data of the DC motor and store it in the specified file for further analysis. This facility makes the instrument suitable for research applications.

Software: Figure 42 shows the flow chart of the DC motor speed control program. The software is written both in 8051 assembly level language and 'C' language. It first initializes the PID parameters, LCD and serial port of microcontroller. After initializing, the program calls ADC subroutine to read the speed of the DC motor through A/D converter. This data of DC motor speed is employed to compute the actual speed in RPM with the help of the equation given below.

$$\text{Speed} = m * (\text{Binary data}) + C$$

Where, 'm' is slope and 'C' is intercept of the straight line drawn between A/D output and rotational speed of the DC motor. The resulting speed of the DC motor is displayed on the LCD module and same will be sent to the host computer through the serial port. The software then subtracts this value from the desired speed (set point speed). The resulting value is called error. This error is applied to PID controller subroutine. The output of the PID is a digital number which is applied to PWM through D/A converter. The output of PWM is series of pulses whose ON-time decides the power applied to the motor through the actuator. Program then enters into the delay subroutine, where it waits for about 1 second. The program then continuously repeats the above procedure to maintain the speed of the DC motor at the desired speed. Software makes the system user friendly. The speed data can be stored in the file name given by the user at the beginning of the program.

The 'C' program and 8086 assembly language program should be linked with help of TCC.EXE program and it should be run on the computer whose serial port is connected to the serial 8051 based dc motor control system. This software reads the DC motor speed through serial port. The software in given in Program-42.

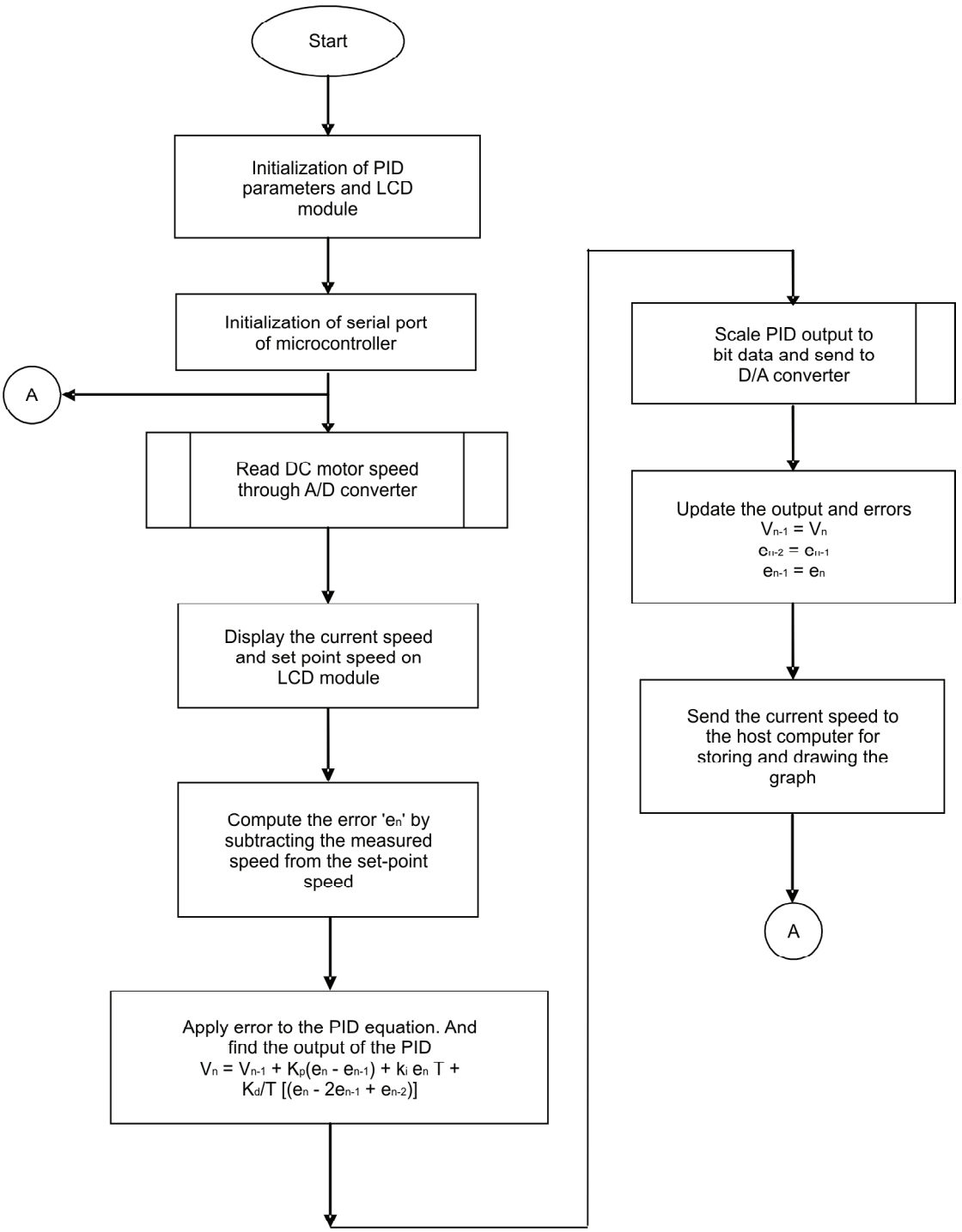
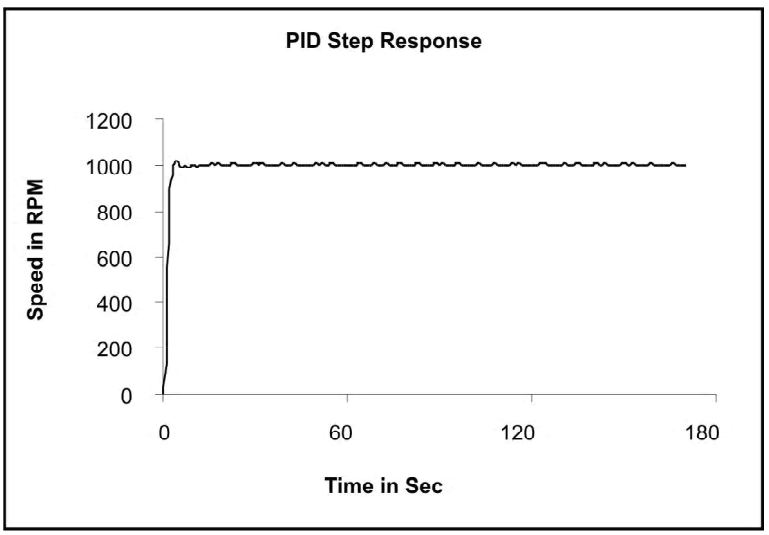
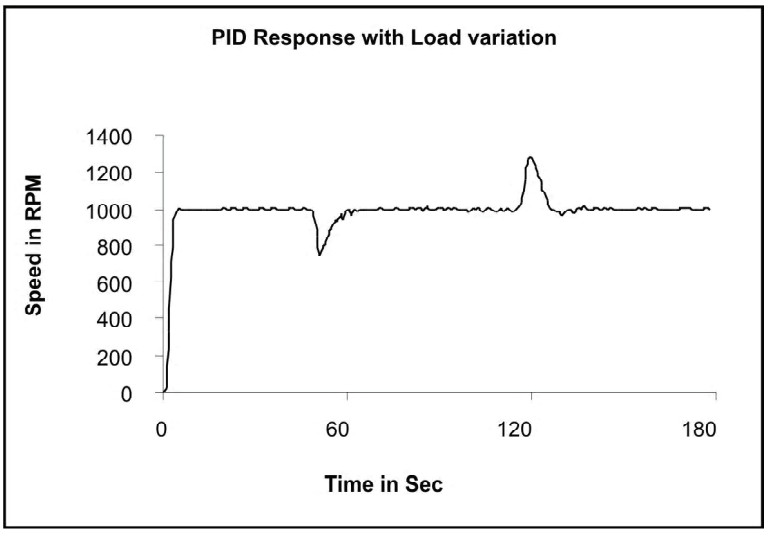

Start
Initialization of PID parameters and LCD module
Initialization of serial port of microcontroller
A
Read DC motor speed through A/D converter
Display the current speed and set point speed on LCD module
Compute the error 'en' by subtracting the measured speed from the set-point speed
Apply error to the PID equation. And find the output of the PID
Vn = Vn-1 + Kp(en - en-1) + ki en T + Kd/T [(en - 2en-1 + en-2)]
Scale PID output to bit data and send to D/A converter
Update the output and errors
Vn-1 = Vn
en-2 = en-1
en-1 = en
Send the current speed to the host computer for storing and drawing the graph
A

Fig. 42. Flow chart of the program.

Results: Figure 43 (a) & (b) shows the speed response of the DC motor for PID step input and load variations respectively. The PID controller gives a good transient as well as steady-state control. It offers rapid proportional response to the error, while having an automatic reset from the integral part to eliminate residual error. The derivative section stabilizes the controller and allows it to respond to the rapid changes or transients in error. Hence, no oscillations are found in the PID response (Fig. 43(a)). After fine tuning, the microcontroller based DC motor speed controller is able to bring the speed to the desired value within 4 Sec and it is able to control the speed of the DC motor with an accuracy of ± 1 rpm.



(a)



(b)

Fig. 43. The speed response of the DC motor for PID (a) step input, and (b) load variations.

Program 42 : Program for DC motor speed control using PID technique

Address	Opcode	Label	Mnemonic	Operand	Comments
		org	0000h		
0000	E4	clr		a	;clear accumulator
0001	F5 A0	mov		p2,a	;clear port-2
0003	12 01 E9	lcall		delay1	;call delay subroutine to ;stop motor initially
					;Initialization of Kp, Ki, Kd, vn,vn_1,en,en_1,en_2,en_3, ref
0006	78 40	mov		r0,#40h	;load address pointer with 40H
0008	74 00	mov		a,#00	
000A	F6	mov		@r0,a	;kp : 40h
000B	08	inc		r0	
000C	F6	mov		@r0,a	; : 41h
000D	08	inc		r0	
000E	F6	mov		@r0,a	;ki : 42h
000F	08	inc		r0	
0010	F6	mov		@r0,a	; : 43h
0011	08	inc		r0	
0012	74 27	mov		a,#27h	
0014	F6	mov		@r0,a	;ref: 44h (2710h = 1000.0 rpm)
0015	08	inc		r0	; : 45h (set point speed)
0016	74 10	mov		a,#10h	
0018	F6	mov		@r0,a	
0019	08	inc		r0	
001A	74 00	mov		a,#0h	
001C	F6	mov		@r0,a	;cur: 46h (current speed in rpm)
001D	08	inc		r0	; : 47h
001E	74 00	mov		a,#0	
0020	F6	mov		@r0,a	
0021	08	inc		r0	
0022	F6	mov		@r0,a	;vn : 48h (current output)
0023	08	inc		r0	
0024	F6	mov		@r0,a	; : 49h
0025	08	inc		r0	
0026	F6	mov		@r0,a	;vn_1: 4ah (previous output)
0027	08	inc		r0	; : 4bh
0028	F6	mov		@r0,a	
0029	08	inc		r0	
002A	F6	mov		@r0,a	;en : 4ch (current error)
002B	08	inc		r0	; : 4dh
002C	F6	mov		@r0,a	
002D	74 00	mov		a,#0	
002F	08	inc		r0	
0030	F6	mov		@r0,a	;en_1: 4eh (previous error)
0031	08	inc		r0	; : 4fh
0032	74 00	mov		a,#0	

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0034	F6	mov		@r0,a	
0035	74 00	mov		a,#0	
0037	08	inc		r0	
0038	F6	mov		@r0,a	;en_2: 50h (previous to previous error)
0039	08	inc		r0	; : 51h
003A	74 00	mov		a,#0	
003C	F6	mov		@r0,a	
003D	74 00	mov		a,#0	
003F	08	inc		r0	
0040	F6	mov		@r0,a	;en_3 : 52h (previous to previous error)
0041	74 00	mov		a,#0	
0043	08	inc		r0	; : 53h
0044	F6	mov		@r0,a	
0045	74 00	mov		a,#0	
0047	08	inc		r0	
0048	F6	mov		@r0,a	;kd : 54h ; (Initialization of Kd)
0049	08	inc		r0	; : 55h
004A	F6	mov		@r0,a	
004B	75 81 40	mov		sp,#40h	;Initialize stack pointer ;with starting address 40h ;initialization of serial port
004E	75 89 20	mov		tmod,#20h	;timer-1 is programmed as 8-bit ;auto reload timer in mode-2
0051	75 88 00	mov		tcon,#00h	;clear TCON register
0054	75 8D F3	mov		th1,#0f3h	;load timer-1 high byte with F3H ;for 1200db
0057	75 98 40	mov		scon,#40h	;serial port is programmed in mode-1
005A	D2 99	setb		ti	;enable transmitter interrupt
005C	D2 8E	setb		tr1	;start timer-1 ;Initialization of LCD display
005E	74 38	mov		a,#38h	;function set control word
0060	12 01 2D	lcall		enable	;send to LCD module
0063	74 0E	mov		a,#0eh	;display ON/OFF control word
0065	12 01 2D	lcall		enable	;send to LCD module
0068	74 06	mov		a,#06h	;entry mode set control word
006A	12 01 2D	lcall		enable	;send to LCD module
006D	74 01	mov		a,#01h	;clear display control word
006F	12 01 2D	lcall		enable	;send to LCD module
0072	74 02	mov		a,#02h	;return home control word
0074	12 01 2D	lcall		enable	;send to LCD module

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0077	12 01 57	lcall		s_display	;call sub-routine to display ;'Speed=' on LCD module
007A	12 01 15	lcall		adc	;call sub-routine for A/D conversion
007D	12 03 16	lcall		mul_1	;call sub-routine to multiply ADC ;data with
0080	12 03 57	lcall		hex_bcd	;call sub-routine to convert HEX ;number into BCD number
0083	12 01 95	lcall		display	;call sub-routine to display speed
0086	74 72	mov		a,#72h	;code for 'r'
0088	12 01 42	lcall		data_disp	;display it on LCD model
008B	74 70	mov		a,#70h	;code for 'p'
008D	12 01 42	lcall		data_disp	;display it on LCD model
0090	74 6D	mov		a,#6dh	;code for 'm'
0092	12 01 42	lcall		data_disp	;display it on LCD model
0095	74 C0	mov		a,#0c0h	;control word to bring cursor ;to the next line in LCD module
0097	12 01 2D	lcall		enable	;send control word to the LCD
009A	12 01 76	lcall		sp_display	;call sub-routine to display ;'Sp_speed=' on LCD module
009D	12 01 C7	lcall		st_speed	;call sub-routine to display ;set-point speed (1000 rpm)
00A0	12 01 AE	lcall		display1	;call sub-routine to display ;it on LCD module
00A3	74 A0	mov		a,#0a0h	;code for blank
00A5	12 01 42	lcall		data_disp	;display it on LCD module
00A8	74 72	mov		a,#72h	;code for 'r'
00AA	12 01 42	lcall		data_disp	;display it on LCD module
00AD	74 70	mov		a,#70h	;code for 'p'
00AF	12 01 42	lcall		data_disp	;display it on LCD module
00B2	74 6D	mov		a,#6dh	;code for 'm'
00B4	12 01 42	lcall		data_disp	;display it on LCD module
00B7	74 02	mov		a,#02h	;control word to bring cursor ;to the left corner of the display
00B9	12 01 2D	lcall		enable	;send control word to the display l1:
00BC	12 01 57	lcall		s_display	;call sub-routine to display ;'Speed=' on LCD module
00BF	12 01 15	lcall		adc	;call sub-routine for A/D ;conversion
00C2	12 03 16	lcall		mul_1	;call sub-routine to multiply ADC ;data with
00C5	12 03 57	lcall		hex_bcd	;call sub-routine to convert HEX ;number into BCD number
00C8	12 01 95	lcall		display	;call sub-routine to display speed
00CB	74 A0	mov		a,#0a0h	;code for blank display
00CD	12 01 42	lcall		data_disp	;display it on LCD module

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
00D0	74 72	mov		a,#72h	;code for 'r'
00D2	12 01 42	lcall		data_disp	;display it on LCD module
00D5	74 70	mov		a,#70h	;code for 'p'
00D7	12 01 42	lcall		data_disp	;display it on LCD module
00DA	74 6D	mov		a,#6dh	;code for 'm'
00DC	12 01 42	lcall		data_disp	;display it on LCD module
00DF	74 C0	mov		a,#0c0h	;control word to bring cursor ;to the next line in LCD module
00E1	12 01 2D	lcall		enable	;send control word to the LCD
00E4	12 01 76	lcall		sp_display	;call sub-routine to display ;'Sp_speed=' on LCD module
00E7	12 01 C7	lcall		st_speed	;call sub-routine to display ;set-point speed (1000 rpm)
00EA	12 01 AE	lcall		display1	;call sub-routine to display ;it on LCD module
00ED	74 A0	mov		a,#0a0h	;code for blank display
00EF	12 01 42	lcall		data_disp	;display it on LCD module
00F2	74 72	mov		a,#72h	;code for 'r'
00F4	12 01 42	lcall		data_disp	;display it on LCD module
00F7	74 70	mov		a,#70h	;code for 'p'
00F9	12 01 42	lcall		data_disp	;display it on LCD module
00FC	74 6D	mov		a,#6dh	;code for 'm'
00FE	12 01 42	lcall		data_disp	;display it on LCD module
0101	12 01 F6	lcall		pid	;call sub-routine for PID
0104	12 02 EF	lcall		dac	;call D/A converter sub-routine
0107	12 03 E2	lcall		serial	;call sub-routine send measured ;speed to the PC through serial port
010A	12 01 D0	lcall		delay	;call delay sub-routine
010D	74 02	mov		a,#02h	;control word to bring cursor ;to the left corner of the display
010F	12 01 2D	lcall		enable	;send control word to the display
0112	02 00 BC	ljmp		l1	;repeat above instructions from ;location 'l1' ;sub-routine for ADC0809
0115	D2 B2	adc:	setb	p3.2	; set ALE and SOC high
0117	00	nop			
0118	00	nop			
0119	00	nop			
011A	00	nop			
011B	00	nop			
011C	00	nop			
011D	00	nop			
011E	00	nop			
011F	00	nop			

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0120	00	nop			
0121	00	nop			
0122	C2 B2	clr		p3.2	;clear ALE and SOC
0124	30 B3 FD	jnb		p3.3, $	;check EOC, if EOC is low ;then poll same line else go to ;read the digital data
0127	E5 80	mov		a,p0	;read the digital data from port-0
0129	F8	mov		r0,a	;store it in r0
012A	79 00	mov		r1,#00h	
012C	22	ret			
				enable:	
012D	F5 90	mov		p1,a	;send control word to display
012F	C2 B4	clr		p3.5	;RS=0
0131	D2 B5	setb		p3.6	;enable=1
0133	75 F0 FF	mov		b,#0ffh	;small delay
0136	D5 F0 FD	djnz		b,$	
0139	C2 B5	clr		p3.6	;enable=0
013B	75 F0 FF	mov		b,#0ffh	;small delay
013E	D5 F0 FD	djnz		b,$	
0141	22	ret			;return to main program
				data_disp:	
0142	F5 90	mov		p1,a	;send character code to display
0144	D2 B4	setb		p3.5	;RS=1
0146	D2 B5	setb		p3.6	;enable=1
0148	75 F0 FF	mov		b,#0ffh	;small delay
014B	D5 F0 FD	djnz		b,$	
014E	C2 B5	clr		p3.6	;enable=0
0150	75 F0 FF	mov		b,#0ffh	;small delay
0153	D5 F0 FD	djnz		b,$	
0156	22	ret			;return to main program
				s_display:	
0157	74 53	mov		a,#53h	;code for ‘S’
0159	12 01 42	lcall		data_disp	;display it on LCD module
015C	74 70	mov		a,#70h	;code for ‘p’
015E	12 01 42	lcall		data_disp	;display it on LCD module
0161	74 65	mov		a,#65h	;code for ‘e’
0163	12 01 42	lcall		data_disp	;display it on LCD module
0166	74 65	mov		a,#65h	;code for ‘e’
0168	12 01 42	lcall		data_disp	;display it on LCD module
016B	74 64	mov		a,#64h	;code for ‘d’
016D	12 01 42	lcall		data_disp	;display it on LCD module
0170	74 3D	mov		a,#3dh	;code for ‘=’
0172	12 01 42	lcall		data_disp	;display it on LCD module
0175	22	ret			;return to main program
				sp_display:	
0176	74 53	mov		a,#53h	;code for ‘S’
0178	12 01 42	lcall		data_disp	;display it on LCD module
017B	74 74	mov		a,#74h	;code for ‘t’
017D	12 01 42	lcall		data_disp	;display it on LCD module
0180	74 5F	mov		a,#5fh	;code for ‘_’
0182	12 01 42	lcall		data_disp	;display it on LCD module
0185	74 53	mov		a,#53h	;code for ‘S’
0187	12 01 42	lcall		data_disp	;display it on LCD module

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
018A	74 70	mov		a,#70h	;code for 'p'
018C	12 01 42	lcall		data_disp	;display it on LCD module
018F	74 3D	mov		a,#3dh	;code for '='
0191	12 01 42	lcall		data_disp	;display it on LCD module
0194	22	ret			;return to main program
0195				display:	
0195	EB	mov		a,r3	;read most significant digit
0196	44 30	orl		a,#30h	;convert it into ASCII
0198	12 01 42	lcall		data_disp	;display it on LCD
019B	EC	mov		a,r4	;read next digit
019C	44 30	orl		a,#30h	;convert it into ASCII
019E	12 01 42	lcall		data_disp	;display it on LCD
01A1	ED	mov		a,r5	;read next digit
01A2	44 30	orl		a,#30h	;convert it into ASCII
01A4	12 01 42	lcall		data_disp	;display it on LCD
01A7	EE	mov		a,r6	;read next digit
01A8	44 30	orl		a,#30h	;convert it into ASCII
01AA	12 01 42	lcall		data_disp	;display it on LCD
01AD	22	ret			;return to main program
					display1:
01AE	E9	mov		a,r1	;read most significant digit
1AF	44 30	orl		a,#30h	;convert it into ASCII
01B1	12 01 42	lcall		data_disp	;display it on LCD
01B4	EA	mov		a,r2	;read next digit
01B5	44 30	orl		a,#30h	;convert it into ASCII
01B7	12 01 42	lcall		data_disp	;display it on LCD
01BA	EB	mov		a,r3	;read next digit
01BB	44 30	orl		a,#30h	;convert it into ASCII
01BD	12 01 42	lcall		data_disp	;display it on LCD
01C0	EC	mov		a,r4	;read next digit
01C1	44 30	orl		a,#30h	;convert it into ASCII
01C3	12 01 42	lcall		data_disp	;display it on LCD
01C6	22	ret			;return to main program
					;sub-routine for set-point speed 1000 rpm
					st_speed:
01C7	79 01	mov		r1,#01h	;load r1,r2,r3,r4 with number 10000
01C9	7A 00	mov		r2,#00h	;to display on LCD module (set point)
01CB	7B 00	mov		r3,#00h	
01CD	7C 00	mov		r4,#00h	
01CF	22	ret			;delay sub-routine

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
				delay:	
01D0	7D 05	mov		r5,#05h	
01D2	7F FF	l8: mov		r7,#0ffh	
01D4	7E FF	l7: mov		r6,#0ffh	
01D6	DE FE	djnz		r6,$	
01D8	DF FA	djnz		r7,l7	
01DA	DD F6	djnz		r5,l8	
01DC		ret			
					;delay sub-routine
					delay2:
01DD	01	mov		r5,#01h	
01DE	7F 0F	l13: mov		r7,#0fh	
01E0	7E FF	l12: mov		r6,#0ffh	
01E2	DE FE	djnz		r6,$	
01E4	DF FA	djnz		r7,l12	
01E6	DD F6	djnz		r5,l13	
01E8	22	ret			
					;large delay sub-routine
01E9				delay1:	
01E9	7D 30	mov		r5,#30h	
01EB	7F FF	l10: mov		r7,#0ffh	
01ED	7E FF	l9 mov		r6,#0ffh	
01EF	DE FE	djnz		r6,$	
01F1	DF FA	djnz		r7,l9	
01F3	DD F6	djnz		r5,l10	
01F5	22	ret			
					;sub-routine for PID program
					;program for finding error 'en = ref - cur'
01F6	pid:				
01F6	78 46	mov		r0,#46h	; 46h current speed
01F8	E6	mov		a,@r0	
01F9	FA	mov		r2,a	
01FA	08	inc		r0	;47h
01FB	E6	mov		a,@r0	
01FC	FB	mov		r3,a	
01FD	78 44	mov		r0,#44h	
01FF	E6	mov		a,@r0	
0200	FC	mov		r4,a	
0201	08	inc		r0	
0202	E6	mov		a,@r0	
0203	C3	clr		c	
0204	9B	subb		a,r3	
0205	FD	mov		r5,a	
0206	EC	mov		a,r4	
0207	9A	subb		a,r2	
0208	FC	mov		r4,a	
0209	78 4C	mov		r0,#4ch	;4ch ref : set point speed
020B	EC	mov		a,r4	
020C	F6	mov		@r0,a	
020D	ED	mov		a,r5	
020E	08	inc		r0	;4dh
020F	F6	mov		@r0,a	

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
		;*** program for finding 'kp = 2*(en-en_1)'***;			
0210	79 02	mov		r1,#2	
0212	78 4E	mov		r0,#4eh	;en_1 : 4eh previous error
0214	E6	mov		a,@r0	
0215	FA	mov		r2,a	
0216	08	inc		r0	;4fh
0217	E6	mov		a,@r0	
0218	FB	mov		r3,a	
0219	78 4C	mov		r0,#4ch	;en : 4ch current error
021B	E6	mov		a,@r0	
021C	FC	mov		r4,a	
021D	08	inc		r0	;4dh
021E	E6	mov		a,@r0	
021F	C3	clr		c	
0220	9B	subb		a,r3	
0221	FD	mov		r5,a	
0222	EC	mov		a,r4	
0223	9A	subb		a,r2	
0224	FC	mov		r4,a	
0225	E9	mov		a,r1	
0226	8D F0	mov		b,r5	
0228	A4	mul		ab	
0229	FD	mov		r5,a	
022A	AE F0	mov		r6,b	
022C	EC	mov		a,r4	
022D	89 F0	mov		b,r1	
022F	A4	mul		ab	
0230	2E	add		a,r6	
0231	FC	mov		r4,a	
0232	78 40	mov		r0,#40h	;kp : 40h load Kp
0234	EC	mov		a,r4	
0235	F6	mov		@r0,a	
0236	ED	mov		a,r5	
0237	08	inc		r0	;41h
0238	F6	mov		@r0,a	
					;program for finding 'ki = 1*en'
0239	79 01	mov		r1,#1	
023B	78 4C	mov		r0,#4ch	;en : 4ch
023D	E6	mov		a,@r0	
023E	FC	mov		r4,a	
023F	08	inc		r0	;4dh
0240	E6	mov		a,@r0	
0241	FD	mov		r5,a	
0242	E9	mov		a,r1	
0243	8D F0	mov		b,r5	
0245	A4	mul		ab	
0246	FD	mov		r5,a	
0247	AE F0	mov		r6,b	
0249	EC	mov		a,r4	

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
024A	89 F0	mov		b,r1	
024C	A4	mul		ab	
024D	2E	add		a,r6	
024E	FC	mov		r4,a	
024F	78 42	mov		r0,#42h	;ki = 42h
0251	EC	mov		a,r4	
0252	F6	mov		@r0,a	
0253	ED	mov		a,r5	
0254	08	inc		r0	;43h
0255	F6	mov		@r0,a	
					;program for finding 'kd = 1*(en-2*en_1+en_2)'
0256	79 01	mov		r1,#1	
0258	78 4E	mov		r0,#4eh	;en_1 : 4eh
025A	E6	mov		a,@r0	
025B	FC	mov		r4,a	
025C	08	inc		r0	;4fh
025D	E6	mov		a,@r0	
025E	FD	mov		r5,a	
025F	2D	add		a,r5	
0260	FB	mov		r3,a	
0261	EC	mov		a,r4	
0262	3C	addc		a,r4	
0263	FA	mov		r2,a	
0264	78 4C	mov		r0,#4ch	;en : 4ch
0266	E6	mov		a,@r0	
0267	FC	mov		r4,a	
0268	08	inc		r0	;4dh
0269	E6	mov		a,@r0	
026A	C3	clr		c	
026B	9B	subb		a,r3	
026C	FD	mov		r5,a	
026D	EC	mov		a,r4	
026E	9A	subb		a,r2	
026F	FC	mov		r4,a	
0270	78 50	mov		r0,#50h	;en_2 : 50h
0272	E6	mov		a,@r0	
0273	FA	mov		r2,a	
0274	08	inc		r0	;51h
0275	E6	mov		a,@r0	
0276	FB	mov		r3,a	
0277	ED	mov		a,r5	
0278	2B	add		a,r3	
0279	FD	mov		r5,a	
027A	EC	mov		a,r4	
027B	3A	addc		a,r2	
027C	FC	mov		r4,a	
027D	E9	mov		a,r1	
027E	8D F0	mov		b,r5	
0280	A4	mul		ab	
0281	FD	mov		r5,a	

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0282	AE F0	mov		r6,b	
0284	EC	mov		a,r4	
0285	89 F0	mov		b,r1	
0287	A4	mul		ab	
0288	2E	add		a,r6	
0289	FC	mov		r4,a	
028A	78 54	mov		r0,#54h	;kd : 54h
028C	EC	mov		a,r4	
028D	F6	mov		@r0,a	
028E	ED	mov		a,r5	
028F	08	inc		r0	;55h
0290	F6	mov		@r0,a	
					;program for finding 'vn = (kp+ki+kd+vn_1)'
0291	78 42	mov		r0,#42h	;ki : 42h
0293	E6	mov		a,@r0	
0294	FA	mov		r2,a	
0295	08	inc		r0	;43h
0296	E6	mov		a,@r0	
0297	FB	mov		r3,a	
0298	78 40	mov		r0,#40h	;kp : 40h
029A	E6	mov		a,@r0	
029B	FC	mov		r4,a	
029C	08	inc		r0	;41h
029D	E6	mov		a,@r0	
029E	2B	add		a,r3	
029F	FD	mov		r5,a	
02A0	EC	mov		a,r4	
02A1	3A	addc		a,r2	
02A2	FC	mov		r4,a	
02A3	78 54	mov		r0,#54h	;kd : 54h
02A5	E6	mov		a,@r0	
02A6	FA	mov		r2,a	
02A7	08	inc		r0	;55h
02A8	E6	mov		a,@r0	
02A9	FB	mov		r3,a	
02AA	ED	mov		a,r5	
02AB	2B	add		a,r3	
02AC	FD	mov		r5,a	
02AD	EC	mov		a,r4	
02AE	3A	addc		a,r2	
02AF	FC	mov		r4,a	
02B0	78 4A	mov		r0,#4ah	;vn_1 : 4ah
02B2	E6	mov		a,@r0	
02B3	FA	mov		r2,a	
02B4	08	inc		r0	;4bh
02B5	E6	mov		a,@r0	
02B6	FB	mov		r3,a	
02B7	ED	mov		a,r5	
02B8	2B	add		a,r3	
02B9	FD	mov		r5,a	
02BA	EC	mov		a,r4	

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
02BB	3A	addc		a,r2	
02BC	FC	mov		r4,a	
02BD	78 48	mov		r0,#48h	;vn :48h
02BF	EC	mov		a,r4	
02C0	F6	mov		@r0,a	
02C1	ED	mov		a,r5	
02C2	08	inc		r0	;49h
02C3	F6	mov		@r0,a	
					;program for updating vn_1 = vn
02C4	78 48	mov		r0,#48h	;vn : 48h
02C6	E6	mov		a,@r0	
02C7	FC	mov		r4,a	
02C8	08	inc		r0	;49h
02C9	E6	mov		a,@r0	
02CA	FD	mov		r5,a	
02CB	78 4A	mov		r0,#4ah	;vn_1 : 4ah
02CD	EC	mov		a,r4	
02CE	F6	mov		@r0,a	
02CF	ED	mov		a,r5	
02D0	08	inc		r0	;4bh
02D1	F6	mov		@r0,a	
					;program for updating en_2 = en_1
02D2	78 4E	mov		r0,#4eh	;en_1 : 4eh
02D4	E6	mov		a,@r0	
02D5	FC	mov		r4,a	
02D6	08	inc		r0	;4fh
02D7	E6	mov		a,@r0	
02D8	FD	mov		r5,a	
02D9	78 50	mov		r0,#50h	;en : 50h
02DB	EC	mov		a,r4	
02DC	F6	mov		@r0,a	
02DD	ED	mov		a,r5	
02DE	08	inc		r0	;51h
02DF	F6	mov		@r0,a	
					;program for updating en_1 = en
02E0	78 4C	mov		r0,#4ch	;en : 4ch
02E2	E6	mov		a,@r0	
02E3	FC	mov		r4,a	
02E4	08	inc		r0	;4dh
02E5	E6	mov		a,@r0	
02E6	FD	mov		r5,a	
02E7	78 4E	mov		r0,#4eh	;en_1 : 4eh
02E9	EC	mov		a,r4	
02EA	F6	mov		@r0,a	
02EB	ED	mov		a,r5	
02EC	08	inc		r0	;4fh
02ED	F6	mov		@r0,a	
02EE	22	ret			
					;program to scale PID output(vn/100) to 8-bit number ;and sending 8-bit number to D/A converter

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
02EF	E5 48	dac: mov		a,48h	
02F1	FA	mov		r2,a	
02F2	E5 49	mov		a,49h	
02F4	F9	mov		r1,a	
02F5	EA	mov		a,r2	
02F6	B4 61 05	cjne		a,#61h,less1	
02F9	7F FF	s_max: mov		r7,#0ffh	
02FB	02 03 12	ljmp		dac_out	
02FE	50 F9	less1: jnc		s_max	
0300	7F FF	mov		r7,#0ffh	
0302	75 F0 64	mov		b,#100	
0305	0F	rep_sb: inc		r7	
0306	E9	mov		a,r1	
0307	C3	clr		c	
0308	95 F0	subb		a,b	
030A	F9	mov		r1,a	
030B	EA	mov		a,r2	
030C	94 00	subb		a,#00h	
030E	FA	mov		r2,a	
030F	50 F4	jnc		rep_sb	
0311	C3	clr		c	
0312	EF	dac_out: mov		a,r7	
0313	F5 A0	mov		p2,a	
0315	22	ret			
					;16X16 multiplication program ;inputs r3.r2 X r1.r0 ;outputs r3_high byte, r2_low byte
				mul_1:	
					;multiplicand in r3_high byte, r2_low byte ;multiplier in r1_high byte, r0_low byte
0316	7A 4E	mov		r2,#4eh	
0318	7B 00	mov		r3,#00h	
031A	7F 00	mul1: mov		r7,#00	
031C	7E 00	mov		r6,#00	
031E	88 F0	mov		b,r0	
0320	EA	mov		a,r2	
0321	A4	mul		ab	
0322	FC	mov		r4,a	
0323	AD F0	mov		r5,b	
0325	89 F0	mov		b,r1	
0327	EA	mov		a,r2	
0328	A4	mul		ab	
0329	2D	add		a,r5	

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
032A	FD	mov		r5,a	
032B	E5 F0	mov		a,b	
032D	3E	addc		a,r6	
032E	FE	mov		r6,a	
032F	50 01	jnc		mul_loop1	
0331	0F	inc		r7	
				mul_loop1:	
0332	88 F0	mov		b,r0	
0334	EB	mov		a,r3	
0335	A4	mul		ab	
0336	2D	add		a,r5	
0337	FD	mov		r5,a	
0338	E5 F0	mov		a,b	
033A	3E	addc		a,r6	
033B	FE	mov		r6,a	
033C	50 01	jnc		mul_loop2	
033E	0F	inc		r7	
				mul_loop2:	
033F	89 F0	mov		b,r1	
0341	EB	mov		a,r3	
0342	A4	mul		ab	
0343	2E	add		a,r6	
0344	FE	mov		r6,a	
0345	E5 F0	mov		a,b	
0347	3F	addc		a,r7	
0348	FF	mov		r7,a	
0349	EC	mov		a,r4	
034A	FA	mov		r2,a ;l_byte	
034B	ED	mov		a,r5	
034C	FB	mov		r3,a ;h_byte	
					;store current speed in 46h & 47h locations
034D	F9	mov		r1,a	
034E	EA	mov		a,r2	
034F	F8	mov		r0,a	
0350	89 46	mov		46h,r1	
0352	88 47	mov		47h,r0	
0354	7A 00	mov		r2,#00h	
0356	22	ret			
					;input 16 bit hex number is in r2_hb,r1,r0_lb
					;output bcd number is in r0_hb,r1,r2,r3,r4,r5,r6,r7_lb
				hex_bcd:	
0357	7F 19	mov		r7,#19h	
0359	12 03 B7	lcall		conv1	
035C	EB	mov		a,r3	
035D	FC	mov		r4,a	
035E	7F 19	mov		r7,#19h	
0360	12 03 B7	lcall		conv1	
0363	EB	mov		a,r3	
0364	FD	mov		r5,a	

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
0365	7F 19	mov		r7,#19h	
0367	12 03 B7	lcall		conv1	
036A	EB	mov		a,r3	
036B	FE	mov		r6,a	
036C	7F 19	mov		r7,#19h	
036E	12 03 B7	lcall		conv1	
0371	EB	mov		a,r3	
0372	FF	mov		r7,a	
0373	EC	mov		a,r4	
0374	54 0F	anl		a,#0fh	
0376	F8	mov		r0,a	
0377	EC	mov		a,r4	
0378	54 F0	anl		a,#0f0h	
037A	C4	swap		a	
037B	F9	mov		r1,a	
037C	ED	mov		a,r5	
037D	54 0F	anl		a,#0fh	
037F	FA	mov		r2,a	
0380	ED	mov		a,r5	
0381	54 F0	anl		a,#0f0h	
0383	C4	swap		a	
0384	FB	mov		r3,a	
0385	EE	mov		a,r6	
0386	54 0F	anl		a,#0fh	
0388	FC	mov		r4,a	
0389	EE	mov		a,r6	
038A	54 F0	anl		a,#0f0h	
038C	C4	swap		a	
038D	FD	mov		r5,a	
038E	EF	mov		a,r7	
038F	54 0F	anl		a,#0fh	
0391	FE	mov		r6,a	
0392	EF	mov		a,r7	
0393	54 F0	anl		a,#0f0h	
0395	C4	swap		a	
0396	FF	mov		r7,a	
0397	EF	mov		a,r7	
0398	F5 08	mov		08h,a	
039A	EE	mov		a,r6	
039B	F5 09	mov		09h,a	
039D	ED	mov		a,r5	
039E	F5 0A	mov		0ah,a	
03A0	EC	mov		a,r4	
03A1	F5 0B	mov		0bh,a	
03A3	E8	mov		a,r0	
03A4	FF	mov		r7,a	
03A5	E9	mov		a,r1	
03A6	FE	mov		r6,a	

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
03A7	EA		mov	a,r2	
03A8	FD		mov	r5,a	
03A9	EB		mov	a,r3	
03AA	FC		mov	r4,a	
03AB	EF		mov	08h	
03AC	F8		mov	r0,a	
03AD	E5 09		mov	a,09h	
03AF	F9		mov	r1,a	
03B0	E5 0A		mov	a,0ah	
03B2	FA		mov	r2,a	
03B3	E5 0B		mov	a,0bh	
03B5	FB		mov	r3,a	
03B6	22		ret		
03B7	C3	conv1:	clr	c	
03B8	74 00		mov	a,#00h	
03BA	FB		mov	r3,a	
03BB	C3	conv2:	clr	c	
03BC	74 00		mov	a,#00h	
03BE	DF 01		djnz	r7, cont	
03C0	22		ret		
				cont:	
03C1	E8		mov	a,r0	
03C2	33		rlc	a	
03C3	F8		mov	r0,a	
03C4	E9		mov	a,r1	
03C5	33		rlc	a	
03C6	F9		mov	r1,a	
03C7	EA		mov	a,r2	
03C8	33		rlc	a	
03C9	FA		mov	r2,a	
03CA	EB		mov	a,r3	
03CB	F5 F0		mov	b,a	
03CD	35 F0		addc	a,b	
03CF	D4		da	a	
03D0	FB		mov	r3,a	
03D1	50 E8		jnc	conv2	
03D3	E8		mov	a,r0	
03D4	34 00		addc	a,#00h	
03D6	F8		mov	r0,a	
03D7	E9		mov	a,r1	
03D8	34 00		addc	a,#00h	
03DA	F9		mov	r1,a	
03DB	EA		mov	a,r2	
03DC	34 00		addc	a,#00h	
03DE	FA		mov	r2,a	
03DF	02 03 BB		ljmp	conv2	
					;program to send the speed to PC through serial port
				serial:	
03E2	E5 46		mov	a,46h	;read high byte of speed from 46h
03E4	D2 B4		setb	p3.4	;set DTR

Program 42 : Contd.

Address	Opcode	Label	Mnemonic	Operand	Comments
03E6	12 03 F1	lcall		ser_out	;call sub-routine to send high byte ;to serial port
03E9	E5 47	mov		a,47h	;read low byte of speed from 47h
03EB	D2 B4	setb		p3.4	;set DTR
03ED	12 03 F1	lcall		ser_out	;call sub-routine to send low ;byte to serial port
03F0	22	ret			;return to main program
				ser_out:	
03F1	30 99 FD	jnb		ti,$	;poll transmitter flag until it goes ;HIGH
03F4	C2 99	clr		ti	;clear transmitter flag
03F6	F5 99	mov		sbuf,a	;send data to the serial port
03F8	C2 B4	clr		p3.4	;clear DTR
03FA	00	nop			;small delay
03FB	00	nop			
03FC	00	nop			
03FD	00	nop			
03FE	22	ret			;return to main program
03FF		end			;program ends

6

Microcontroller Based Temperature Control System

AIM: To design and construct microcontroller based temperature control system.

COMPONENTS: 89C51, A/D converter ICL7135, transducer Pt100, precision voltage reference LM329, instrumentation amplifier INA101, 2-5.1V Zener diodes, 100k Pot, 0.1µF, 0.47µF, 1.0µF metalized polyester capacitors (one each), 1.0µF tantalum capacitor, 10µF capacitor, 100kΩ multi-turn pot, 4–100kΩ, 8.2kΩ,100Ω, 27Ω resistors, 12.0 MHz crystals, switch, LCD module, bread board and connecting wires.

EQUIPMENT: Personal computer, CRO.

PRINCIPLE: The block diagram of microcontroller based temperature control system is shown in Fig. 44. The temperature of the steel body is sensed with a sensor Pt-100 and the output of the sensor is suitably modified with the help of a signal conditioner. The output of the signal conditioner is an analog voltage which is converted into a digital data with the help of an analog to digital (A/D) converter which is interfaced to the microcontroller. The microcontroller reads the digital data (current temperature) from A/D converter and computes the error by subtracting the measured temperature from the desired temperature (set-point). Then microcontroller solves the PID equations. The output of control program is a digital data, which is fed to D/A converter. The output of D/A converter is an analog voltage which is given to the pulse width modulator (PWM) circuit. The 'ON' time of the pulse width modulator changes in accordance with the D/A converter output. The output of the PWM is given to the actuator (MOSFET). The actuator regulates the quantity of heat energy added to or removed from the strip heater connected to the steel body to maintain it at the desired temperature.

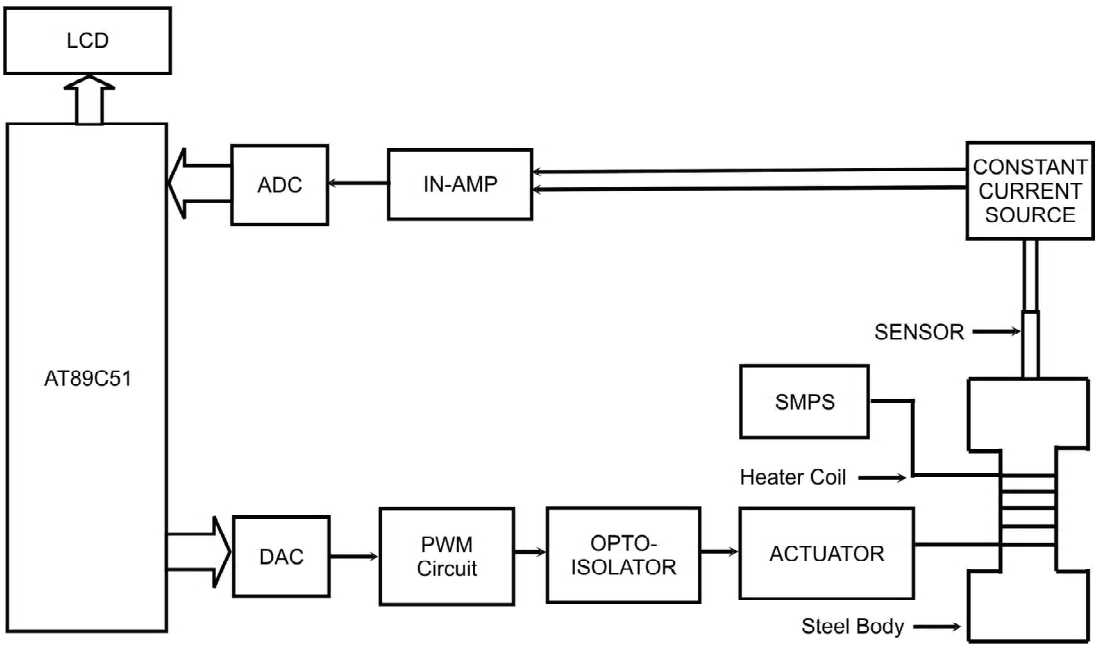


Fig. 44. Block diagram of microcontroller based temperature control system.

Hardware: Figure 45 shows the complete circuit schematic of microcontroller based temperature control system. The following sections deal with the different sub-systems of the system.

a. Temperature Sensor and Signal Conditioning Circuit

The conventional temperature transducers have some significant limitations like non-linearity, low output, narrow range etc. The platinum resistance thermometer (Pt-100) is used for the present study to overcome the above difficulties. It operates on the principle of change in electrical resistance of platinum wire as a function of temperature. The relation between temperature and resistance of the platinum wire is given by the relation (i)

$$R_T = R_o (1 + AT + BT^2 + \ldots\ldots) \sim R_o (1 + AT) \quad \ldots\ldots \text{(i)}$$

where all the second and higher order terms are negligibly small.

Figure 40 shows the complete circuit diagram of the system. Here, the signal conditioning circuit is employed for converting resistance changes of Pt-100 sensor into voltage changes. A constant current source has been designed using operational amplifier. This eliminates the lead-wire error of Pt-100. A stable voltage source is constructed using LM329 which gives a constant voltage of 6.9V. This reference voltage

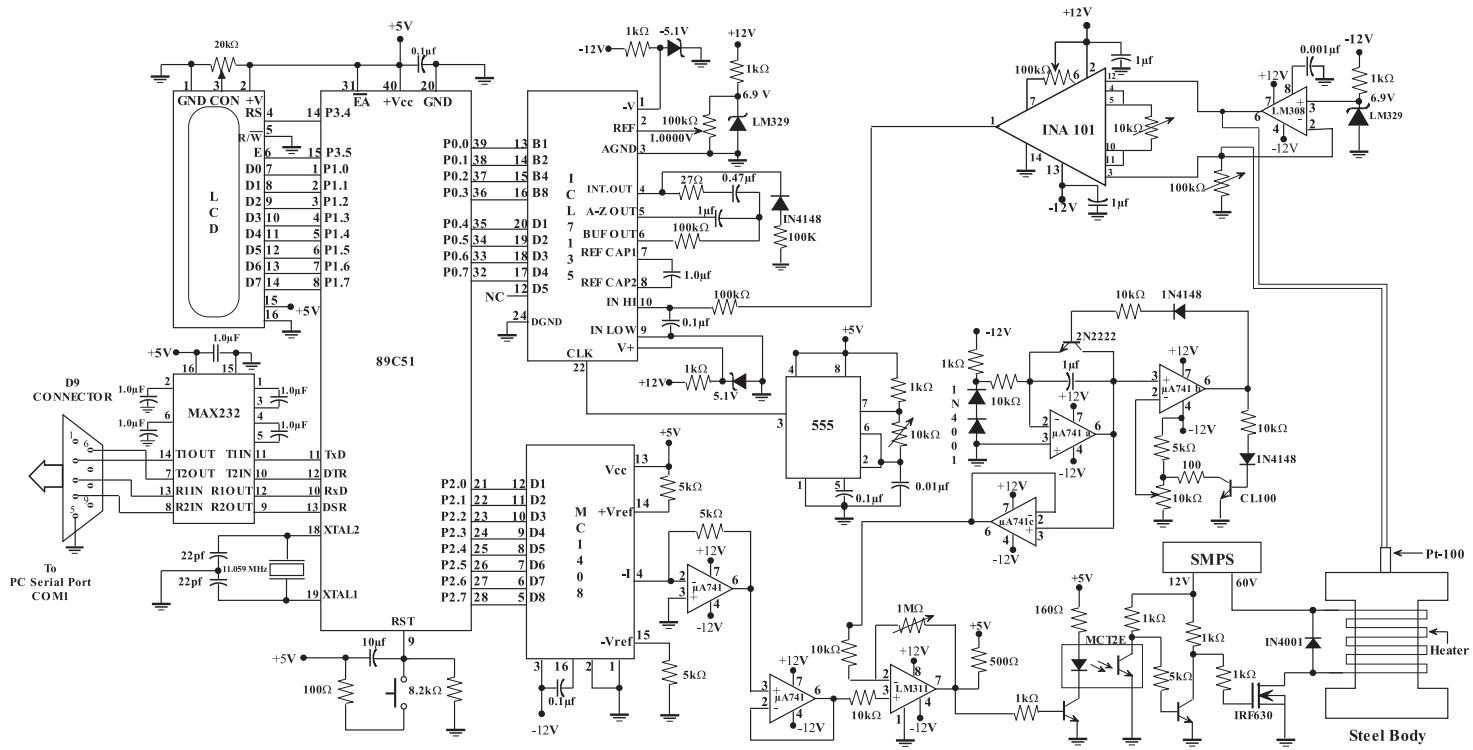


Fig. 45. Circuit schematic of microcontroller based temperature control system.

is applied to the non-inverting input of A1. The Pt-100 acts as a feedback resistance and the resistance R_S determines the amount of current through Pt-100. Since, the potential difference between input terminals of the operational amplifier is zero. Hence, the voltage at the inverting terminal is also equal to 6.9V. Therefore, the current flowing through the resistance (R_S) is equal to $6.9/R_S$. Since, the input impedance of the amplifier is very high, the current flowing through Pt-100 is equal to the current flowing though resistance R_S. In the present study a constant current source is designed for 100 mA, the resistance R_S can be calculated by the equation (ii)

$$R_S = (6.9V)/100 \text{ mA} = 69.0 \text{ K} \quad \ldots\ldots\ldots \text{(ii)}$$

A 100 K multi-turn potentiometer is used as R_S, and it is adjusted to the value 69.9 K. This gives a constant current of 100mA. When this current passes through the temperature sensor Pt-100, produces a differential voltage given by the relation (iii)

$$\Delta V = 100 \text{ A} * (\text{Resistance of Pt-100}) \quad \ldots\ldots\ldots \text{(iii)}$$

this voltage is amplified through an instrumentation amplifier INA101 to increase the sensitivity of the system. This amplified voltage is a measure of the resistance of sensor Pt-100 and is used for computing the temperature of the steel body through software making use of the resistance to temperature conversion relation ($y_i=m_ix_i+c_i$) and microcontroller displays the temperature directly in °C.

b. Interfacing of A/D converter ICL7135 with Microcontroller

The output of the instrumentation amplifier is given to the 4½ digit A/D converter. The output of the A/D converter ICL 7135 is in multiplexed BCD form. The BCD data are output from the converter on lines B1, B2, B4, and B8. A logic high is output on one of the digit strobe lines D5, D4, D3, D2, and D1 to indicate when BCD code for corresponding digit is placed on BCD outputs. The data from this A/D converter are read by polling the bit corresponding to a strobe line (until that bit goes high) and store the data in reserved memory locations for future reference. After reading the BCD code for one digit, the bit corresponds to the strobe line for the next digit is poled until that bit goes high and is stored in next memory location. The process is repeated until the data for all the 5 digits are acquired. Figure 45 shows an interface between an A/D converter with microcontroller.

c. Interfacing of LCD module with Microcontroller

The circuit diagram shown in Fig. 45 presents the interfacing of the display module with the microcontroller. The data bus D0-D7 of display module is connected to P1.0-P1.7 of microcontroller and the control pins RS and E are connected to the P3.4 and P3.5 pins of 8751 respectively. The command word is sent to the command

register by making RS LOW through P3.4 and the data can be sent to the display RAM by making RS HIGH through P3.4. A HIGH pulse of about 40 micro second duration is sent to the pin E of the display module to fetch/send the data/command word to the display, which is through the pin P3.5 of 8751. The display module pin R/W is connected to the ground. The third pin of the display is connected to the variable terminal of the potentiometer which varies the contrast of the display.

d. Interfacing of D/A converter and PWM with Microcontroller

The D/A converter MC1408 is interfaced with the microcontroller through port-2. The output of D/A converter is connected to the non-inverting terminal of comparator (LM311) and inverting input of comparator is connected to the output of ramp generator circuit which is designed using three op-amps (741a-c) as shown in Fig. 45. The 'ON' time of the comparator depends on the output of the D/A converter. If the output of the D/A converter is large, then 'ON' time of the comparator output is more, similarly if the output of the D/A converter is small, then 'ON' time of the comparator output is small.

e. Microcontroller

The role assigned to the microcontroller in the present study is to acquire the temperature data, evaluate the error by subtracting it from a reference (set-point), solve the control algorithms and finally apply the output of the algorithms to the D/A converter. The output of the D/A converter changes the 'ON' time of the PWM. The later controls the energy supplied to the heater through a power MOSFET. To enable the microcontroller to implement the PI, PID and improved PID control algorithms, the continuous differential equations are to be converted to a discrete difference equations as given below. The velocity algorithms are given by the equations

$$V_n = V_{n-1} + K_p(e_n - e_{n-1}) + K_i\, e_n\, T \qquad \ldots\ldots\ldots \text{(iv)}$$

$$V_n = V_{n-1} + K_p(e_n - e_{n-1}) + K_i\, e_n\, T + K_d/T\,[(e_n - 2e_{n-1} + e_{n-2})] \qquad \ldots\ldots\ldots \text{(v)}$$

The above equation (v) is a standard PID difference equation. At any instant of time the current value of the PID output V_n is calculated based on the previous value of the PID output V_{n-1}, current error e_n, previous error $e_{n-1,}$ $e_{n-2,}$ are previous to the previous errors, the cycle time T and weighing constants (K_p, K_i, K_d). The output of control program is a digital value, which is applied to the D/A converter.

f. Serial port

The MAX232 is also incorporated in the microcontroller board for serial communication with the host computer. This enables the user to tune PID parameters and can change

the set-point through this port. The computer can read the temperature data of the steel body and store it in the specified file for further analysis. This facility makes the instrument suitable for research applications.

g. Working of Microcontroller based PID Controller

Figure 46 shows the photographs of microcontroller board and total system. After sampling the temperature of the steel body through A/D converter, the microcontroller compares this value with the desired temperature (set value). The difference between these two values is applied to the PID algorithm and the solution of the algorithm is fed to D/A converter, whose output is connected to the PWM input. The 'ON' time of the PWM depends on the output of the D/A converter. If the output of the D/A converter is large, then 'ON' time of the PWM is more and if the output of the D/A converter is small, then 'ON' time of the comparator is small. Presenting a small digital number to the D/A converter produces small analog output voltage, this produces small 'ON' time and results in supply of lesser output energy to the heater. Conversely a large digital number to the D/A converter produces large analog output causing a large 'ON' time. Hence, more energy is applied to the heater. The output of PWM is connected to MOSFET through the opto-isolator which controls the power to the heater. This variation in duty cycle controls the power

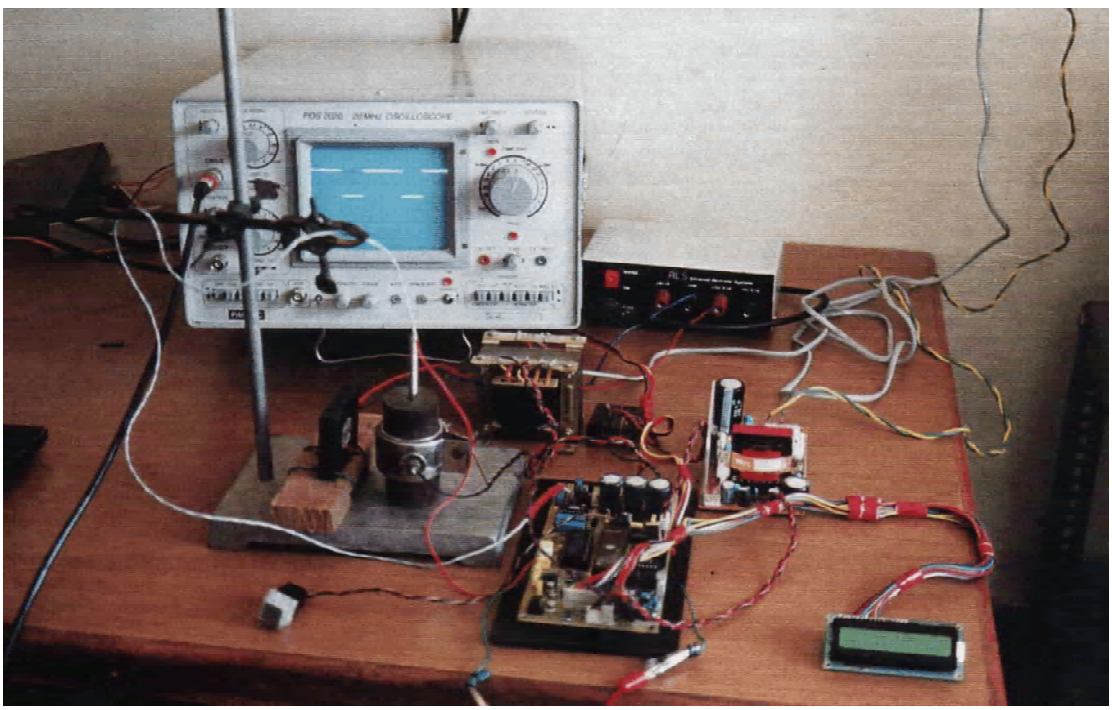


Fig. 46. Photograph of temperature control system.

applied to the power MOSFET and hence the amount of energy supplied to the heater. Thus, the later maintains the temperature of the steel body at the desired value. The PID temperature controller designed by the authors found to regulate the temperature of the steel body with in ±0.1°C.

Software: The software is written in 8051 assembly language. Figure 47 shows the flow chart of the temperature measurement and control program. The software first initializes PID parameters (Kp, Ki, Kd) and LCD module. After initialization the program then calls the A/D conversion subroutine to read the resistance of the temperature sensor Pt-100 through analog to digital converter ICL7135. This data of temperatures are employed to compute the temperature with the help of the equation given below in relation

$$\text{temp} = m * (\text{Binary data}) + c \qquad \text{..... (vii)}$$

where 'm' = slope and 'c' is intercept of the straight line. The resulting temperature will be displayed on the LCD module and sent to the host computer through the serial port. After displaying temperature, the software then subtracts this temperature from the desired temperature. The resulting number is called the error. This error is applied to a PID algorithm. The output of PID is a digital number, which is given to the D/A converter. Then it enters into the delay subroutine, where it waits for about 2 seconds. This compensates for the lag time of the steel body. The program continuously repeats the above procedure to maintain the temperature of the steel body at the desired value.

Software makes the system user friendly. It provides the on-line tuning of PID parameters through host computer. The temperature data can be stored in the file name given by the user at the beginning of the program or at any time. Software also provides on-line variation of set-point, which facilitates the system to study the step variation response. The embedded C software is given in Program-43.

Results and Discussion: Figure 48a-c shows the temperature response of the steel body for PI, PID and improved PID controllers respectively. The response of PI controller (Fig. 48a) is oscillatory and it takes more time to stabilize. The PID controller gives a good transient as well as steady-state control. It offers rapid proportional response to error, while having an automatic reset from the integral part to eliminate residual error. The derivative section stabilizes the controller and allows it to respond to the rapid changes or transients in error. Hence, no oscillations are found in the PID response (Fig. 48b). Fig. 48c shows the set-point variation response of improved PID controller. The improved PID controller response is better than normal PID controller response in terms of less raise time and less settling time. After fine tuning, the microcontroller based temperature controller is able to bring the temperature of the steel body to the desired value within 15 minutes. And it is able to control the temperature of the steel body within ±0.1°C.

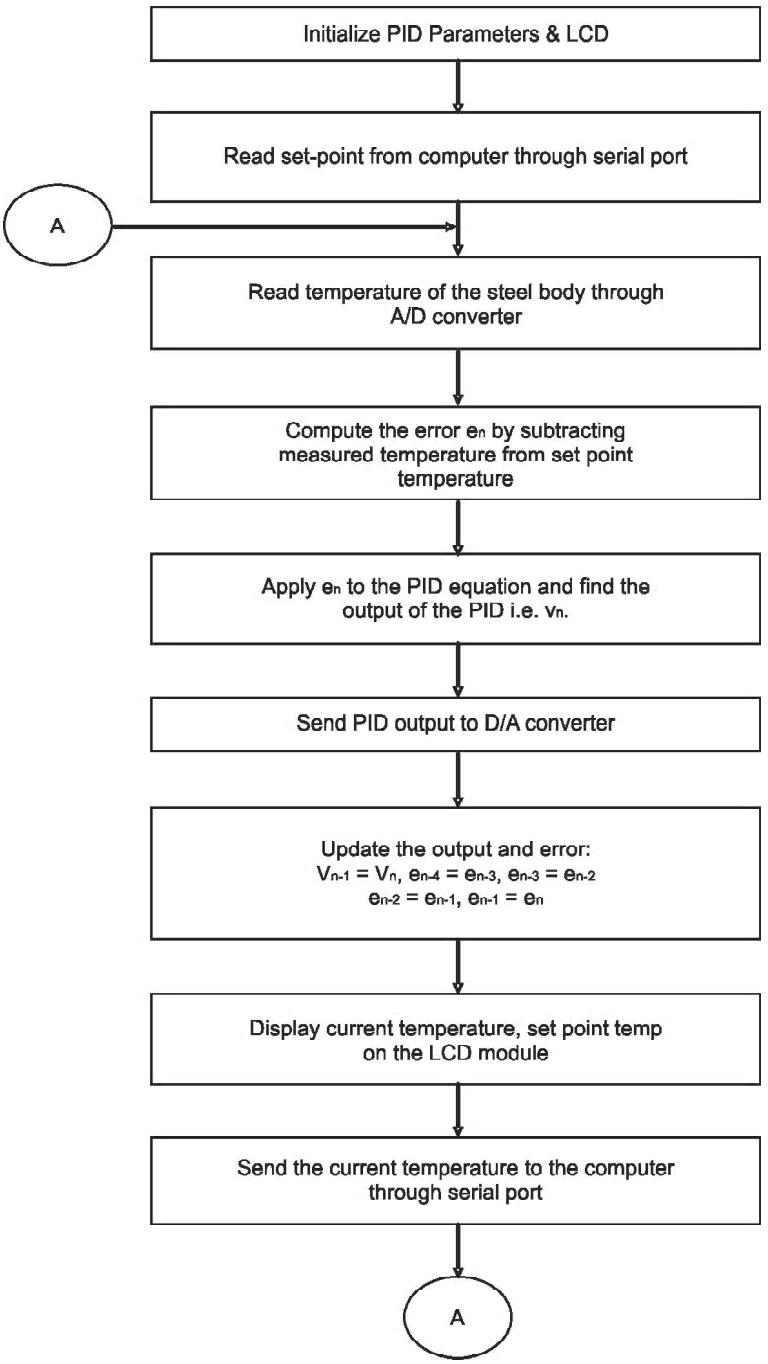
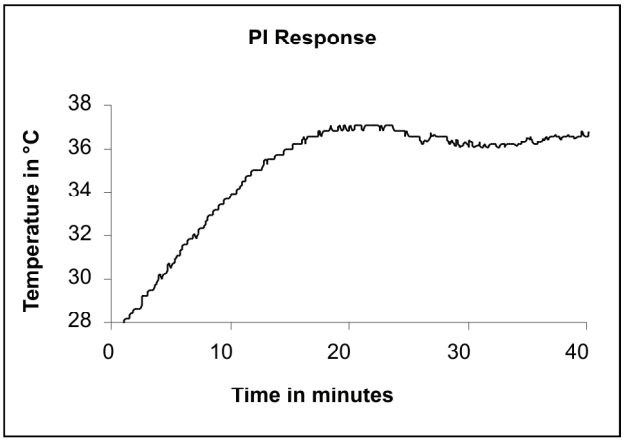
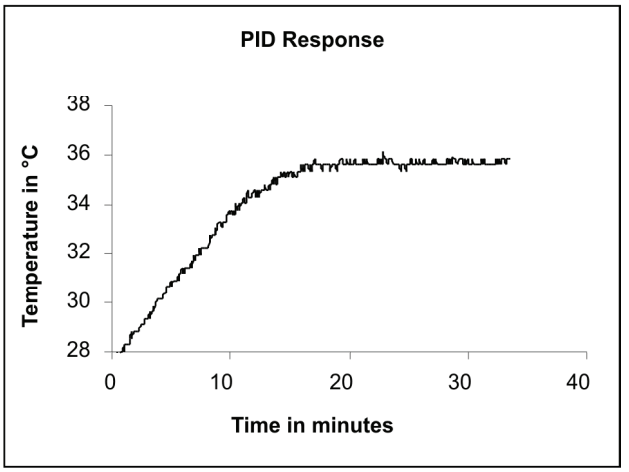

Initialize PID Parameters & LCD
Read set-point from computer through serial port
A
Read temperature of the steel body through A/D converter
Compute the error e_n by subtracting measured temperature from set point temperature
Apply e_n to the PID equation and find the output of the PID i.e. v_n.
Send PID output to D/A converter
Update the output and error:
$V_{n-1} = V_n$, $e_{n-4} = e_{n-3}$, $e_{n-3} = e_{n-2}$
$e_{n-2} = e_{n-1}$, $e_{n-1} = e_n$
Display current temperature, set point temp on the LCD module
Send the current temperature to the computer through serial port
A

Fig. 47. Flow chart of temperature control program.





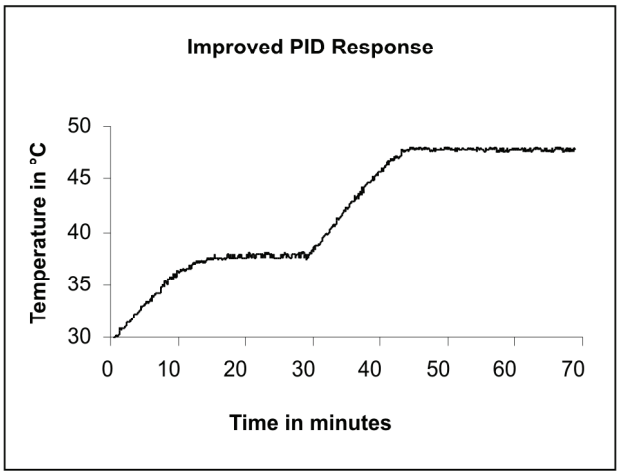


Fig. 48a-c. The temperature response of the steel body for PI, PID and improved PID controllers.

Program 43

```
/* Program for PID temperature control system */

#include <stdio.h>
#include <reg51.h>

   sbit   x2 = P0^7;
   sbit   x3 = P0^6;
   sbit   x4 = P0^5;
   sbit   x5 = P0^4;

   sbit    rs = P3^4;
   sbit     e = P3^5;
   sbit   DTR = P3^2;
   sbit   DSR = P3^3;

   int d1,d2,d3,d4,d5,temp;

   void main (void)

   {
   int adc_7135(void);
   int lcd_comand();
   int lcd_data();
   int lcd_int(void);
   void dac(int);
   void ser_init(void);
   int serial_in(void);
   void serial_out(int);

   float v1,kp=2.0,ki=1.0,kd=1.0,t=1.0,sp_temp,cur_temp;
   float en,en_1=0.0,en_2=0.0,vn,vn_1=0.0;
   int i;
   int sp[] = {'S','e','t','_','T','e','m','p','='};
   int ct[] = {'C','u','r','_','T','e','m','p','='};
   int  k[] = {'o','C'};

   ser_init();       //initialization of serial port
   lcd_int();        //initialization of LCD module

   P1 = 0x02;        //bring cursor to the left corner
   lcd_comand();     //of the display
```

```
    for(i=0;i<9;i++) //display 'Cur_Temp=' on LCD module
  {
  P1 = ct[i];
  lcd_data();
  }

  v1 = adc_7135();                //read current temperature
//through ADC
  v1 = (2.618*v1)+10.39 - 273.0; //compute temperature
  v1 = v1*100;
  temp = (int)v1;
  conv();        //un-pack the number
  display();     //display on LCD module

  for (i = 0;i<2;i++) //display 'oC' on LCD module
  {
  P1 = k[i];
  lcd_data();
  }

  serial_out(temp);    //send current temperature
                       //to the PC through serial port

  for(i=0;i<9;i++)     //display 'Set_Temp=' on LCD module
  {
  P1 = sp[i];
  lcd_data();
  }
  sp_temp = serial_in();    //read set point temperature
                            //from the PC through serial
                            //port
  temp = sp_temp;
      conv();               //un-pack the number
  display();

  for (i = 0;i<2;i++)       //display 'oC' on LCD module
  {
  P1 = k[i];
  lcd_data();
  }
  sp_temp = sp_temp/100;
  en = sp_temp-cur_temp;    //compute error
```

```
 /* compute PID output */
    vn = vn_1 + kp*en + ki*(en - en_1)*t +
              (kd/t)*(en - 2*en_1 + en_2);

 /* update previous error and previous to previous error
     and previous output */
    en_2 = en_1;
    en_1 = en;
    vn_1 = vn;
   rep:
   P1 = 0x02;          //bring cursor to the left corner
   lcd_comand();       //of the display

    for(i=0;i<9;i++)    //display 'Cur_Temp=' on LCD module
   {
   P1 = ct[i];
   lcd_data();
   }
    v1 = adc_7135();  //read current temperature
//through ADC
    v1 = (2.618*v1)+10.39 - 273.0; //compute temperature
    v1 = v1*100;
    temp = (int)v1;
    conv();
    display();

   for(i=0;i<2;i++)     //display 'oC' on LCD module
   {
   P1 = k[i];
   lcd_data();
   }
   en = sp_temp-cur_temp; //compute error

 /* compute PID output */
    vn = vn_1 + kp*en + ki*(en - en_1)*t +
              (kd/t)*(en - 2*en_1 + en_2);

 /* update previous error and previous to previous error
     and previous output */
    en_2 = en_1;
    en_1 = en;
    vn_1 = vn;
```

```
    dac(vn);          //scale out of PID and send to DAC
    goto rep;         //repeat above steps continuously
   }
/* function for initialization of serial port */
void ser_init(void)
   {
   TMOD = 0x20;
   TCON = 0x00;
    TH1 = 0xFD;
   SCON = 0x40;
      TI = 1;
      RI = 1;
   }
/*function for analog to digital conversion*/
int adc_7135(void)
   {
   int v;
   do{
   d1 = P0;
   d1 = d1 & 0xf0;
   }while (d1!=0);

   d1 = P0 & 0x01;

   while (x2!=1);
   d2 = P0 & 0x0f;

   while (x3!=1);
   d3 = P0 & 0x0f;

   while (x4!=1);

   d4 = P0 & 0x0f;

   while (x5!=1);
   d5 = P0 & 0x0f;

   v = d1*10000+d2*1000+d3*100+d4*10+d5;
   return v;
   }
/* function for converting packed BCD number into unpacked BCD
number */
   conv()
   {
```

```
   d1 = temp/1000;

   d2 = temp % 1000;
   d2 = d2/100;

   d3 = temp % 100;
   d3 = d3/10;

   d4 = temp % 10;
   }
/*function for displaying temperature on LCD module */
   display()
   {
   d1 = d1 | 0x30;
   P1 = d1;
   lcd_data();

   d2 = d2 | 0x30;
   P1 = d2;
   lcd_data();

   P1 = 0x2e;
   lcd_data();

   d3 = d3 | 0x30;
   P1 = d3;
   lcd_data();

   d4 = d4 | 0x30;
   P1 = d4;
   lcd_data();
   }

/*scaling of output of PID and applying to DAC */
   void dac(int cn)
   {
   int dac_out;
   if(cn = 0x6100)
   dac_out = 0xff;

   else
   dac_out = cn/100;
```

```
   P2 = dac_out;
   }
/* function for receiving the set point speed through
    serial port of PC */
   serial_in()
   {
   int lb,hb,s_data_in;

   REN = 1;
   while(DSR == 0);
   while(RI != 1);
   RI = 0;
   lb = SBUF;

       REN = 1;
   while(DSR == 0);
   while(RI != 1);
   RI = 0;
   hb = SBUF;

       s_data_in = hb*256+lb;
       return s_data_in;
   }
/* function for sending the current speed to the PC through
    serial port */

   void serial_out(int s_data_out)
   {
   int hb,lb;
   hb = s_data_out/256;
   lb = s_data_out % 256;

   DTR = 1;
   while(TI != 1);
   TI = 0;
   SBUF = hb;
   DTR = 0;

   DTR = 1;
   while(TI != 1);
   TI = 0;
   SBUF = lb;
   DTR = 0;
   }
```

```
/* function for initializtion of LCD module */
   int lcd_int()
   {
   P1 = 0x38;
   lcd_comand();

   P1 = 0x0e;
   lcd_comand();

   P1 = 0x06;
   lcd_comand();

   P1 = 0x01;
   lcd_comand();

   P1 = 0x02;
   lcd_comand();
   }

/* function for sending control word to the LCD module */
   int lcd_comand()
   {
   int i;
   rs = 0;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
/* function for sending data to the LCD module */
   int lcd_data()
   {
   int i;
   rs = 1;
   e =  1;
   for (i = 1;i<1000;i++);
   e = 0;
   for (i = 1;i<1000;i++);
   }
```

Experiments with Microcontrollers

This book addresses the experimental needs of students/researchers/teachers of various disciplines in science, engineering, and technology on microcontrollers at the graduate and post-graduate levels.

One of the most widely produced and used microcontrollers in the world is the 8051 model and is the most popular to study. Hence, the Intel 8051/89C51 microcontroller is chosen to design the experiments/projects. The basic knowledge of 8051 microcontroller, viz., architecture and its instruction set required to understand the experiments is given in the book.

This book is organised into three parts. Part I deals with the software experiments – addition, subtraction, multiplication, code conversion, etc. All the software programs are written in assembly language. Part II describes the hardware experiments. Some of the hardware experiments are designed with microcontroller kits and many are discussed with 89C51 microcontroller alone. Every experiment is written in such a way that an experimenter need not require additional information to perform the experiment. In most of the experiments, both assembly language and embedded 'C' programs are given. Part III deals with the complete projects. These projects are related to the applications of microcontroller either for measuring and/or controlling a parameter. The salient features and descriptions of sensors/transducers, signal conditioning elements and actuators are provided wherever it is necessary.

P. Bhaskar is Associate Professor, Dept. of Instrumentation Technology, Gulbarga University P.G. Centre, Raichur, Karnataka.

K. Malakondaiah is Professor, Dept. of Instrumentation, Sri Krishnadevaraya University, Anantapur, Andhra Pradesh.

dreamtech PRESS

978-93-89872-49-1